UNIVERSIDAD DE MÁLAGA
E.T.S.I. INFORMÁTICA
Dpto. Lenguajes y Ciencias de la Computación

Tesis Doctoral

EMERGENT OPTIMIZATION: DESIGN AND APPLICATIONS IN TELECOMMUNICATIONS AND BIOINFORMATICS

José Manuel García Nieto

Málaga, 2013

UNIVERSIDAD
DE MÁLAGA

LENGUAJES Y
CIENCIAS DE LA
COMPUTACIÓN
UNIVERSIDAD DE MÁLAGA

Emergent Optimization: Design and Applications in Telecommunications and Bioinformatics

Ph.D. Thesis Dissertation in Computer Sciences

Author
José Manuel García-Nieto

Supervisor
Dr. Enrique Alba Torres

Department of
Lenguajes y Ciencias de la Computación

UNIVERSITY OF MÁLAGA

February 2013

Departamento de Lenguajes y Ciencias de la Computación
Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga

El Dr. D. **Enrique Alba Torres**, Catedrático de Universidad perteneciente al Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga,

Certifica

que, D. **José Manuel García Nieto**, Ingeniero en Informática por la Universidad de Málaga, ha realizado en el Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga, bajo su dirección, el trabajo de investigación correspondiente a su Tesis Doctoral titulada:

Emergent Optimization: Design and Applications in Telecommunications and Bioinformatics

Revisado el presente trabajo, estimo que puede ser presentado al tribunal que ha de juzgarlo. Y para que conste a efectos de lo establecido en la legislación vigente, autorizo la presentación de esta Tesis Doctoral en la Universidad de Málaga.

En Málaga, febrero del 2013

Fdo: Dr. Enrique Alba Torres

Emergent Optimization: Design and Applications in Telecommunications and Bioinformatics

Ph.D. Thesis Dissertation in Computer Sciences
written by
José Manuel García-Nieto

and supervised by
Prof. Enrique Alba Torres

February 2013

A mis padres, Juan y Clotilde,
por la educación y confianza
que depositaron en mi

A Carmen Jesús y a Pedro,
por su apoyo, cariño y paciencia
durante todo este tiempo

Agradecimientos

Antes de empezar esta memoria, me gustaría expresar mi agradecimiento tanto a las personas como a las instituciones que han contribuido de una forma u otra a la realización de esta tesis doctoral.

En primer lugar, como no puede ser de otra manera, agradezco sinceramente a mi director, el Doctor Enrique Alba Torres, que haya confiado en mi capacidad y que me haya guiado en la realización de este trabajo, así como en otras muchas cosas de la vida, siempre ofreciéndome todo su tiempo, dedicación y amistad.

También me gustaría agradecer, de manera muy especial, a mis compañeros del grupo de investigación NEO, que han compartido conmigo tantas horas de trabajo en el laboratorio. A Paco, Francis, Gabriel y Bernabé, que me acogieron desde el primer momento y me ayudaron con su enorme experiencia. A Guillermo, Juanjo y Briseida, que me animaron siempre y compartieron conmigo tantos problemas, alegrías y sentimientos en este arduo camino del doctorado. Por supuesto, agradezco también a Javier, Jamal, Sebastián y Daniel, por su apoyo diario y a los cuales animo en sus propias tesis y andaduras profesionales. No me puedo olvidar de Antonio y de Juan Miguel, que siempre se han prestado a ayudarme y a resolver mis dudas de manera desinteresada, así como de todos los que han pasado por el laboratorio: Juan, Dani, Julián y Javier, sin querer dejar de mencionar a ninguno pues a todos agradezco. Para mí es un verdadero lujo formar parte de este fantástico grupo reunido por Enrique.

Quiero agradecer también a todos los compañeros "extranjeros", de tantos países, que me han acompañado en mi trabajo y me han mostrado lo diverso, grande y bonito que es el mundo. De manera muy especial, agradezco a mis más estrechos colaboradores: los argentinos Javier Apolloni y Ana Carolina Olivera, así como a Pablo, Sergio, Andreas y tantos otros.

Por supuesto, debo expresar mi más sincero agradecimiento al Profesor El-Ghazali Talbi por haberme facilitado mis estancias doctorales en el INRIA-Lille en Francia, durante tres interesantes meses de mi vida en los que tanto aprendí.

Termino estas líneas dando gracias, de todo corazón, a mi esposa Carmen Jesús y a mi hijo Pedro, que siempre están a mi lado apoyándome y dando luz a mis días. Gracias de todo corazón también a mis queridas hermanas, Cloti, Nati y Carmen, que me han acompañado durante toda mi vida. Por último, gracias a mis maravillosos padres, Juan y Clotilde, que me lo han dado todo y me han enseñado que con trabajo, voluntad y esfuerzo puedo conseguir cualquier cosa que me proponga.

Acknowledgements

This work thesis has been partially funded by the Spanish Ministry of Sciences and Innovation (MEC) and the European Regional Development Fund (FEDER), under contract TIN2008-06491-C04-01 of the Mstar project (http://mstar.lcc.uma.es). It has been also partially funded by the National (MEC) RoadMe TIN2011-28194 (http://roadme.lcc.uma.es) and regional (Junta de Andalucía) DIRICOM P07-TIC-03044 (http://diricom.lcc.uma.es) projects. Finally, the author is supported by a FPI grant with code BES-2009-018767 from MEC.

Contents

Acknowledgements **iv**

I Motivation and Fundamentals **1**

1 Introduction **3**
1.1 Motivation . . . 3
1.2 Objectives and Phases . . . 6
1.3 PhD Thesis Contributions . . . 6
1.4 Thesis Organization . . . 7

2 Fundamentals of Metaheuristics **11**
2.1 Metaheuristics . . . 11
2.1.1 Classification of Metaheuristics . . . 15
2.1.2 Extended Models . . . 19
2.1.3 Statistical Validation Procedure . . . 23

3 Fundamentals of Particle Swarm Optimization (PSO) **25**
3.1 PSO: Introduction . . . 25
3.1.1 Canonical PSO . . . 26
3.1.2 Standard Versions of PSO . . . 27
3.1.3 Prominent Versions of PSO . . . 30
3.1.4 Related Approaches: Differential Evolution (DE) . . . 36
3.2 A General Survey on PSO Applications . . . 37
3.2.1 Benchmarking . . . 37
3.2.2 Real World Applications . . . 40

II Algorithm Proposals and Validation on Benchmarks **43**

4 DEPSO: Hybrid PSO with DE **45**
4.1 Introduction . . . 45
4.2 The Algorithm: DEPSO . . . 46
4.3 Experiments on MAEB'09 . . . 47
4.3.1 MAEB'09: Results and Analysis . . . 48
4.4 Experiments on BBOB'09 . . . 52

4.4.1 GECCO'09: Numerical Experiments on Noiseless Functions . . . 53
4.4.2 GECCO'09: Numerical Experiments on Noisy Functions . . . 53
4.5 Run Time Analysis . . . 56
4.6 Conclusions . . . 56

5 RPSO-vm: Velocity Modulation PSO for Large Scale Optimization 59
5.1 Introduction . . . 59
5.2 The Algorithm: RPSO-vm . . . 60
5.3 Experimental Setup . . . 62
5.3.1 Benchmark Functions . . . 62
5.3.2 Parameter Settings . . . 63
5.4 Analysis of Results . . . 63
5.4.1 RPSO-vm Numerical Results . . . 63
5.4.2 Scalability Analysis . . . 66
5.4.3 Computational Effort . . . 70
5.5 Conclusions . . . 71

6 SMPSO: Speed Modulation PSO for Multi-objective Optimization 73
6.1 Introduction . . . 73
6.2 The Basic MOPSO . . . 74
6.3 Studied Approaches . . . 75
6.4 Experimentation . . . 76
6.4.1 Parameterization . . . 76
6.4.2 Methodology . . . 76
6.5 Computational Results . . . 77
6.6 Discussion . . . 79
6.7 Conclusions . . . 83

7 PSO6: Quasi-optimally Informed PSO 85
7.1 Introduction . . . 85
7.2 The Quest for an Optimal Number of Informants . . . 87
7.3 Experimental Setup . . . 87
7.4 Analysis and Discussion . . . 88
7.4.1 Impact of the Number of Informants . . . 88
7.4.2 Performance Comparisons . . . 90
7.4.3 Computational Effort . . . 92
7.4.4 Influence of the Swarm Size . . . 93
7.4.5 Influence of the Problem Dimension . . . 93
7.5 Evolvability Analysis . . . 94
7.5.1 Evolvability Measures . . . 95
7.5.2 Fitness-Distance Analysis . . . 97
7.5.3 Fitness-Fitness Analysis . . . 98
7.6 PSO6 with Multiple Trajectory Search . . . 99
7.6.1 The Proposal: PSO6-Mtsls . . . 99
7.6.2 Experiments with PSO6-Mtsls . . . 101
7.6.3 PSO6-Mtsls: Performance Comparisons . . . 103
7.7 Conclusions . . . 110

III Real World Applications **113**

8 Gene Selection in DNA Microarrays **115**
8.1 Introduction . . . 115
8.2 DNA Microarrays . . . 117
8.3 PMSO for Gene Selection . . . 118
8.4 Experimental Results . . . 120
8.4.1 Microarray Datasets and Data Preprocessing . . . 120
8.4.2 Experimental Settings . . . 121
8.4.3 Performance Analysis . . . 122
8.4.4 Speedup Analysis . . . 124
8.4.5 PMSO versus Parallel Island Genetic Algorithm . . . 125
8.4.6 PMSO versus Other Approaches . . . 126
8.4.7 Biological Analysis of the Results . . . 127
8.5 Conclusions . . . 128

9 Optimizing Software Communication Protocols **129**
9.1 Introduction . . . 129
9.2 Literature Overview . . . 131
9.3 Problem Overview . . . 131
9.3.1 Vehicular Data Transfer Protocol . . . 131
9.4 Optimization Strategy . . . 133
9.5 Experiments . . . 134
9.5.1 Instances: VANET Scenarios . . . 134
9.5.2 Parameter Settings . . . 135
9.5.3 Results and Comparisons . . . 136
9.5.4 Performance Analysis . . . 137
9.5.5 Scalability Analysis . . . 138
9.5.6 QoS Analysis . . . 140
9.6 Conclusions . . . 141

10 Optimal Signal Light Timing **143**
10.1 Introduction . . . 143
10.2 Literature Overview . . . 144
10.3 SUMO: Simulator of Urban MObility . . . 147
10.3.1 SUMO Data Structure . . . 147
10.4 PSO for Traffic Light Scheduling . . . 149
10.4.1 Solution Encoding . . . 149
10.4.2 Fitness Function . . . 150
10.4.3 Optimization Strategy . . . 150
10.5 Methodology of Our Study . . . 152
10.5.1 Instances . . . 152
10.5.2 Experimental Setup . . . 154
10.6 Analysis and Discussion of Results . . . 155
10.6.1 Performance Analysis of Algorithms . . . 156
10.6.2 Comparison with Other Metaheuristics . . . 160
10.6.3 Scalability Analysis . . . 161
10.6.4 Computational Effort . . . 162

10.6.5 Analysis of Solutions . . . 162
10.7 Conclusions . . . 164

IV Conclusions and Future Work **167**

11 Conclusions and Future Work **169**
11.1 Global Conclusions . . . 169
11.2 Future Lines of Research . . . 172

V Appendices **175**

A Publications Supporting this PhD Thesis Dissertation **177**

B Resumen en Español **183**
B.1 Introducción . . . 183
B.2 Organización de la Tesis . . . 184
B.3 Fundamentos . . . 186
B.3.1 Metaheurísticas . . . 186
B.3.2 Particle Swarm Optimization . . . 187
B.4 Propuestas Algorítmicas y Estudios Experimentales . . . 189
B.4.1 Algoritmo Híbrido DEPSO . . . 189
B.4.2 PSO con Modulación de Velocidad: Test de Escalabilidad . . . 190
B.4.3 PSO con Modulación de Velocidad: Versión Multiobjetivo . . . 191
B.4.4 Estudio del Número Óptimo de Informadores en PSO . . . 192
B.4.5 Nueva Versión: PSO6 con MTS . . . 193
B.5 Aplicación a Problemas Reales . . . 193
B.5.1 Selección de Genes en Microarrays de ADN . . . 193
B.5.2 Configuración Óptima de Protocolos en VANETs . . . 194
B.5.3 Programación Óptima de Ciclos en Semáforos . . . 195
B.6 Conclusiones . . . 196

List of Tables **198**

List of Figures **201**

List of Algorithms **203**

Index of Terms **204**

Bibliography **206**

Part I

Motivation and Fundamentals

Chapter 1

Introduction

1.1 Motivation

One of the most important aspects in Computer Sciences Research consists in the analysis and design of optimization algorithms for solving new and complex problems in an efficient way. The main target in this field is then the development of new optimization methods able to solve the aforementioned complex problems with a progressively lower computational effort, and therefore improve the current best methods. As a consequence, the new algorithms allow scientists not only tackling current problems in an efficient way, but also facing tasks that were unsolvable in the past because of their prohibitive computational costs. In this context, researching on exact, *ad hoc* heuristic, and metaheuristic techniques has attracted a lot of attention nowadays. The reason is the increasing number of complex problems that are appearing in the industry and, at the same time, the availability of new better computational resources, such as efficient multicore cluster computers, cloud computing execution platforms, and other high performance environments.

The main advantage of exact algorithms is that they can guarantee the finding of an optimal solution for any problem. However, despite an special emphasis still exists on that, there is a wealth of problems for which there are not exact algorithms able of computing an optimal solution in polynomial time, since the needed computing effort grows exponentially along with the problem dimension. Such a group of problems is usually referred to as NP-hard. In contrast, ad hoc heuristics are usually very fast in problem resolution, although obtaining in general moderate quality solutions. Another drawback in Ad hoc heuristics is that they are not easy to define for certain problems where a minimum knowledge (about the problem) is required, but not available, as commonly happens in black box problem optimization, and usually not allowing cross-fertilization to other similar tasks, what is seen as an unstructured and effort loss in many applications. Metaheuristics on the contrary provide high quality solutions (many times the optima) in an acceptable computing time, allowing structured proposal of techniques and easing cross-fertilization to other domains (additional benefits). As a consequence, Industry (and not only Science) is more and more conscious of these latter techniques, which have been used in a wide range of different applications.

In the context of metaheuristics, *Swarm Intelligence (SI)* approaches are becoming very popular in the last years. SI techniques try to model the emergent behavior of simple collective agents searching as a whole, with the aim of inducing the global learning procedure on the solution of complex problems. In concrete, *Particle Swarm Optimization (PSO)* is possibly the most popular algorithm in the family of SI procedures, since a huge amount of research publications concerning

this technique have appeared since its original design by Kennedy and Eberhart in 1995 [KE95]. PSO is inspired by the sociological focus that collective learning and intelligence can originate from human (or gregarious animal) social interactions within the living environment where they participate. The algorithm is based on the concept that individuals refine their knowledge about the search landscape through social interactions. Optimization patterns emerge when individuals tend to imitate their prominent peers, then making up the intrinsic learning procedure. Each individual, known as a *particle* in PSO, follows two basic decision rules: going back to the best stage visited so far (*individualism*), and imitating the best particle found in its neighborhood (*conformism*).

Particle swarm optimization shows a series of advantages that have given it increasing popularity in the field of computational optimization. Between these advantages, the most relevant ones are based on the facts that:

- it is easy to describe and implement [KE01],
- it does not requires extra computational effort for its own operation [KE01],
- it can be easily modified to manage different problem representations [MCP07],
- there exists a number of available implementations in different programming languages [PCG11],
- it requires few parameters to be tuned [ES00, Tre03],
- it usually performs a fast convergence to successful solutions [CK02],
- it shows an accurate performance on separable problems [LQSB06], and
- it has been used to tackle multitude of industrial real world problems [PKB07, SM09].

Although numerous studies have been carried out concerning the particle swarm optimization algorithm, there is still a big room for research on this topic. In the first place, PSO based algorithms still show a moderate performance prompted by premature convergence on complex problems with several concrete features: non-separable, shifted, rotated, multimodal, and with deceiving landscapes. A number of past works have attempted to mitigate these issues, although with limited success for several reasons: they did not address the cause but they addressed the effect, they were not duly compared with the current state of the art (just with other older PSO versions), they did not use correct benchmark of functions to evaluate the new proposals (in most of cases with few problems, with single local optima at the origin of coordinates), and they did not use statistical procedures to validate their comparisons.

A way of dealing with these issues is to analyze the internal behavior of PSO by using runtime evolvability measures with the aim of better understanding the learning procedure induced by particles. Another way is to put the PSO to stress and scalability tests in order to identify its main limitations on hard conditions. In this way, it is possible to design new operators and to hybridize with other methods that lead the new proposals to reach a high degree of success on a wide range of optimization problems. In this sense, working with numerous and heterogeneous problem functions (and with different dimensions) to cover as much as possible the specific landscape features that directly affect the PSO's performance (non-separability, rotation, shifting, etc.) is a must in order to evaluate the new proposals. In the second place, real world applications are more and more complex and, frequently, involve pre/post processing operations or the use of software simulators that increase the time required for evaluating a candidate solution. Then, the design of advanced PSO models able of computing accurate solutions in a fast manner is to be faced as a

final stress test. A way of fulfilling that requirement is to make use of parallelism for speeding-up the execution time of PSO. Another way is to develop new more advanced PSO versions able of computing accurate representation of solutions in a bounded number of function evaluations.

Our motivation in this thesis is twofold. These steps are in fact part of the present PhD Thesis plan. First, we are interested in designing new Particle Swarm Optimization proposals that solve or mitigate the main disadvantages present in this algorithm. To this point, we have used a series of methodologies to analyze the internal behavior of this algorithm and to identify the main problems or improvement opportunities appearing in existing PSO versions. In order to assess the effectiveness of the new proposals, we have performed comparative studies from two main points of view: solution quality and scalability in terms of the problem size (decision variables). For this task we have followed specific experimental procedures of standard benchmark test suites (CEC'05, SOCO'10, DTLZ, etc.), and we have compared against the most prominent metaheuristics in current state of the art. Second, we are aimed at solving real world complex problems with PSO based algorithms to determine the adaptability of this algorithm to different representations and scenario conditions, within limited computational time, and requiring huge data base management. In concrete, we have focused in this thesis on three NP-Hard real applications trying to cover quite different industry fields: Gene Selection in DNA Microarrays (Bioinformatics), Communication Protocol Tuning in VANETs (Telecommunications), and Signal Lights Timing in traffic management (Urban Mobility).

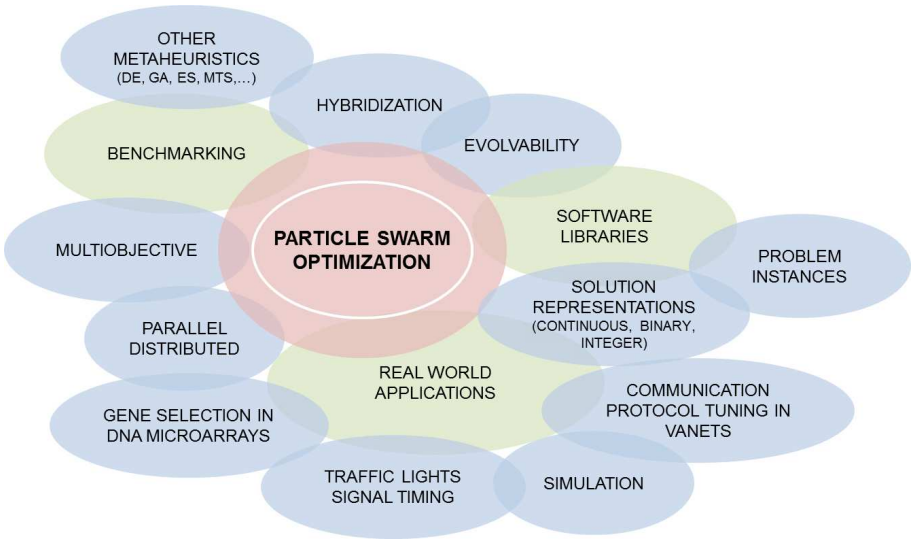


Figure 1.1: Conceptual cloud involving the PSO that we have covered in this PhD Thesis

As a global target of this PhD Thesis, we are interested on investigating the Particle Swarm Optimization from a wide spectrum of algorithmic aspects, with the aim of determining the general purpose capabilities of this algorithm. Figure 1.1 illustrates the conceptual cloud involving the PSO that we have covered in this thesis. Finally, as a product, all that work has also led the development of a software in the scope of MALLBA library, aimed at supporting the design and development of metaheuristics. In addition, a number of realistic problem instances, simulation rule files, software scripts, and free web sites have been generated and are available.

1.2 Objectives and Phases

This thesis considers the analysis of the Particle Swarm Optimization, as well as the design and implementation of new proposals based on this algorithm. This work also addresses the resolution of complex optimization problems with PSO in the domains of: DNA Microarrays, VANETs Communication Protocols, and Signal Lights Timing Programs. These general objectives can be detailed into more specific goals as follows:

- Analyze current particle swarm optimization versions and identify the most important deficiencies it shows on complex optimization.
- Design of new PSO proposals by means of alternative velocity formulations, adapted mechanisms, hybridization with other techniques, and with multi-objective focus.
- Analyze evolvability, scalability, and efficiency issues in existing and new PSO versions.
- Validate our findings in previous points for solving real-world problems.
- Develop innovative approaches that enhance the performance of current PSO, and other optimization techniques, either from the perspective of the quality of the solutions produced, or from the perspective of the computational effort required to reach them. Demonstrate their effectiveness through statistically assessed experimental evaluation.

In order to fulfil these thesis objectives, the work has been carried out as follows. Firs, we have surveyed the concepts of metaheuristics and, in concrete, particle swarm optimization. Then, we have focused on identifying deficiencies of PSO on different problem's landscape characterizations and scalability conditions. After this, we have proposed advanced design issues with optimized learning procedures, with new operators, and hybridizing with other techniques, giving rise to improved versions of PSO (actually to new kinds of algorithms). Secondly, we have applied PSO, as well as other optimization algorithms, for solving the three aforementioned real-world problems taken from different disciplines of Engineering.

1.3 PhD Thesis Contributions

The contributions of this thesis are mainly devoted to the research field of Swarm Intelligence. However, additional interesting contributions can be also highlighted in the research fields of DNA microarrays, communication protocols in VANETs, and in traffic management with signal lights programming. These contributions can be summarized as follows:

- Thorough review and development of Canonical and Standards of PSO (2006, 2007, and 2011), current versions (FIPS, CLPSO, wPSO, GPSO, etc.), and new proposals made in the scope of the MALLBA library of metaheuristics.
- Analysis of PSO and DE basic operations and proposal of DEPSO, a hybrid PSO that makes use of DE operators.
- Proposal and scalability analysis of RPSO-vm, a restarting PSO with velocity modulation procedure.
- Analysis of different variants of multi-objective PSO. Proposal of SMPSO, a multi-objective PSO with velocity modulation.

- Sensitive analysis of the number of informants involved in the learning procedure of PSO. A number of six informants (PSO6) is suggested as the new standard base algorithm.
- Proposal of PSO6-Mtsls, a particle swarm with optimized informants that is hybridized with MTS-LS1, Multiple Trajectory Search local search.
- Comparative analysis of all proposed techniques with the current state of the art, in the context of standard benchmark frameworks (CEC'05, SOCO'10, BBOB'09, DTLZ, etc.).
- Proposal of a parallel extension of Geometric PSO for Gene selection in DNA Microarrays. Evaluation on real gene expression of cancer data sets.
- Realistic modeling of representative problems: Tuning of Communication Protocols in VANETs and Signal Lights Timing Programming, both in urban areas. Evaluation of standard and integer encoding PSO versions on the resolution of these two problems.
- Creation of large size real-world instances on urban scenarios for Tuning of Communication Protocols in VANETs and Signal Lights Timing Programs. These instances were designed as simulation components to the well-known communication and traffic simulators *ns2* and SUMO, respectively.

In addition, a number of scientific articles have been published during the years in which this thesis has been developed that support and validate the impact of these contributions on the scientific community and literature. These publications have appeared in impact journals and fora, and have been subjected to peer review by expert researchers, summing up more than 38 research papers: 10 journal articles indexed by ISI-JCR (3 of them under review), 2 international journal articles, 5 book chapters and papers in LNCS Series, and 19 conference papers (references to these publications can be found in Appendix A).

1.4 Thesis Organization

This thesis work has a double orientation: from algorithmics and towards the application domain, and this reflects into its structure as a document. Thus, this volume is divided into five parts. In the first part we present the fundamentals and basis for the work, metaheuristics as a global family of resolution techniques, the particle swarm optimization as particular algorithm to study, and the methodology that we have employed for assessing and validate the numerical results. The second part is devoted to the analysis, design, and evaluation of our proposals. Comparisons with the state of the art in the scope of standard benchmarks are also performed in this part. The third part is devoted to the problems tackled in this thesis: models, mathematical/computational formulations, instances and results are described and validated. Literature concerning the problem domains is also reviewed in this part. The fourth part recaps the main conclusions drawn throughout the work and gives global comments. Finally, the fifth part contains two appendices concerning the publications derived from this thesis work, as well as a summary in Spanish language.

- **Part I. Motivations and Fundamentals**
  - Chapter 2 introduces the main concepts in the research field of Optimization and Metaheuristics, including classifications and emphasizing Swarm Intelligence algorithms. In the last subsection, the standard statistical validation procedure employed in all the experiments is detailed.

  - Chapter 3 is devoted to present the Particle Swarm Optimization in detail, describing canonical and standard versions of PSO, giving theoretical aspects of its different formulations, enumerating a set of prominent versions of this algorithm, as well as other related techniques. A final part of this chapter concentrates on the usual standard benchmarks of functions employed for the experimentation phases in the following chapters.

- **Part II. Algorithm Proposals and Validation**

  - Chapter 4 addresses the hybridization of PSO with DE to propose the DEPSO algorithm. This chapter starts by describing the new algorithm formulation, and directly goes to the experimental procedure applied to evaluate its performance. In this sense, an extensive set of functions were used consisting on: CEC'05 benchmark test suite, and noisy/noisiless functions of BBOB'09. A series of analysis concerning the different function's features and comparisons against other algorithms are then performed.
  - Chapter 5 presents the Restarting PSO with Velocity Modulation (RPSO-vm), designed to tackle with large scale continuous optimization problems. A scalability tests is applied to this proposal in the scope of special issue of SOCO'10. A series of comparisons and analysis from the point of view of the computational effort are also carried out.
  - Chapter 6 enumerates the most prominent multi-objective PSO (MOPSO) versions and evaluate them on three well-known problem families. After this, our new proposal, Speed Modulation PSO (SMPSO), is then described and evaluated with regards to these previous MOPSO versions and NSGAII.
  - Chapter 7 contains one of the most interesting investigations performed in this thesis. The chapter starts by empirically analyzing the specific number of informant particles that can provide the PSO with an optimized learning procedure, on a big number of different problem functions. After this, an runtime analysis from the point of view of the evolvability is performed with the aim of shedding some light on why certain number of informants (around six) is the best option in the formulation of PSO. Finally, the resulted baseline method PSO6 is hybridized with MTS to generate a new candidate proposal PSO6-Mtsls, aimed at constituting the current state of the art on an extensive benchmark of problem functions (CEC'05+SOCO'10).

- **Part III. Real World Applications**

  - Chapter 8 presents the problem of Gene Selection in DNA Microarrays, and the Parallel Multi-Swarm Optimization (PMSO) algorithm proposed to solve it. This algorithm is based on the binary version of Geometric PSO (GPSO), which has been proven to be specially well adapted to the feature selection problem. The effectiveness of PMSO is analyzed on four well-known public datasets, discovering new and biologically challenging gene subsets, and identifying specific genes that our work suggests as significant ones. Comparisons with several recent state of art methods show the effectiveness of our results in terms of computational time/effort, reduction percentage, and classification rate. A further biological analysis validates our results, since they independently confirm the relevant genes also found in reference works in Biology.
  - Chapter 9 addresses the application of PSO to the efficient Tuning of Communication Protocols in VANETs. The chapter starts with a survey of the literature, and the problem formulation related to the faced VDTP communication protocol. Several analysis and comparisons are then performed from the point of view of the network QoS and the

problem scalability, on different realistic scenarios. Resulted protocol configurations are also compared against human experts ones on real applications.

- Chapter 10 introduces the problem of Signal Lights Timing Programs in vehicular traffic urban environments. After the literature overview and the problem formulation, the optimization strategy is presented, what includes an integer encoding PSO coupled with SUMO simulator. Two traffic instances have been generated for the cities of Malaga and Bahía Blanca. Performance comparisons are carried out from the point of view of fitness quality, and on different scaling scenarios. Resulted timing programs are favourably compared against the ones using human expert information.

- **Part IV. Conclusions**

  - Chapter 11 contains a global review of the thesis work, and revisits the main conclusions drawn. The thesis objectives and main contributions are then discussed in view of the results obtained. Lastly, the future lines of research are briefly sketched and discussed.

- **Part V. Appendices**

  - Appendix A presents the set of related works that have been published during the years in which this thesis work has been carried out.
  - Appendix B is a summary of this volume in Spanish language.
  - The remaining appendices contain: the list of tables, the list of figures, the list of algorithms, the index of terms, and the bibliography.

Chapter 2

Fundamentals of Metaheuristics

2.1 Metaheuristics

This PhD Thesis will develop in the domain of advanced algorithms and metaheuristics in particular [Gol89, Glo03]. A metaheuristic is a high level technique or algorithm for solving complex optimization problems. They are stochastic algorithms which do not guarantee to obtain the optimal solution of a given problem, but when properly tuned allow to obtain near-optimal solutions, often the optimal one, with bounded computation effort.

We will proceed in several steps, from the problem to the techniques to solve it. First, let us give a formal definition of an optimization problem. Assuming, without loss of generality, a *minimization* case, the definition of an *optimization problem* is as follows:

Definition 2.1.1 (Optimization problem)**.** *An optimization problem is defined as a pair* (S, f)*, where* $S \neq \emptyset$ *is called the solution space (or search space), and* f *is a function named* objective function *or* fitness function*, defined as:*

$$f : S \to \mathbb{R} \ . \tag{2.1}$$

Thus, solving an optimization problem consists in finding a solution $i^* \in S$ *such that:*

$$f(i^*) \leq f(i), \quad \forall \ i \in S \ . \tag{2.2}$$

Note that assuming either maximization or minimization does not restrict the generality of the results, since an equivalence can be made between the two cases in the following manner [Gol89, B̈96]:

$$\max\{f(i)|i \in S\} \equiv \min\{-f(i)|i \in S\} \ . \tag{2.3}$$

Depending on the domain which S belongs to, we can speak of *binary* ($S \subseteq \mathbb{B}^*$), *integer* ($S \subseteq \mathbb{N}^*$), *continuous* ($S \subseteq \mathbb{R}^*$), or *heterogeneous* optimization problems ($S \subseteq (\mathbb{B} \cup \mathbb{N} \cup \mathbb{R})^*$).

A simple classification of the optimization methods used throughout the history of Computer Science is shown in Figure 2.1. In a first approach, the techniques can be classified into *Exact* and *Approximate*. Exact techniques, which are based on the mathematical finding of the optimal solution, or an exhaustive search until the optimum is found, guarantee the optimality of the

obtained solution. However, these techniques present some drawbacks. The time they require, though bounded, may be very large, especially for NP-hard problems. Furthermore, it is not always possible to find such an exact technique for every problem. This makes exact techniques not to be a good choice in many occasions, since both their time and memory requirements can become unreasonably high for large scale problems. For this reason, approximate techniques have been widely used by the international research community in the last few decades. These methods sacrifice the guarantee of finding the optimum in favor of providing some satisfactory solution within reasonable times.

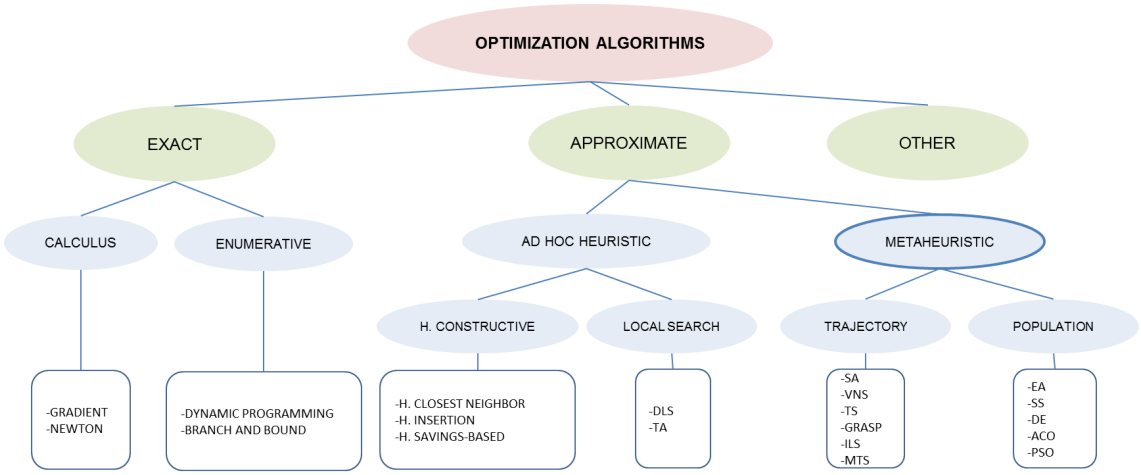


Figure 2.1: General classification of optimization techniques

Among approximate algorithms, we can find two types: *ad hoc* heuristics, and metaheuristics. We focus this chapter on the latter, although we mention before *Ad hoc* heuristics, which can in turn be divided into *constructive heuristics* and *local search methods*. Constructive heuristics are usually the swiftest methods. They construct a solution from scratch by iteratively incorporating components until a complete solution is obtained, which is returned as the algorithm output. Finding some constructive heuristic can be easy in many cases, but the obtained solutions are of low quality in general since they use simple rules for such construction. In fact, designing one such method that actually produces high quality solutions is a nontrivial task, since it mainly depends on the problem, and requires thorough understanding of it. For example, in problems with many constraints it could happen that many partial solutions do not lead to any feasible solution.

Local search or gradient descent methods start from a complete solution. They rely on the concept of *neighbourhood* to explore a part of the search space defined for the current solution until they find a *local optimum*. The neighbourhood of a given solution s, denoted as $N(s)$, is the set of solutions (neighbours) that can be reached from s through the use of a specific modification operator (generally referred to as a *movement*). A local optimum is a solution having equal or better objective function value than any other solution in its own neighbourhood. The process of exploring the neighbourhood, finding and keeping the best neighbour, is repeated in a process until the local optimum is found. Complete exploration of a neighbourhood is often unapproachable, therefore some modification of this generic scheme has to be adopted. Depending on the movement operator, the neighbourhood varies and so does the manner of exploring the search space, simplifying or complicating the search process as a result.

During the 70's, a new class of approximate algorithms appeared whose basic idea was to combine operations in a structured (family-like) way in a higher level to achieve an efficient and effective search of the problem landscape. These techniques are called *metaheuristics*. The term was first introduced by Glover [Glo86], and until it was ultimately adopted by the scientific community, these techniques were named *modern heuristics* [Ree93]. This class of algorithms includes many diverse techniques such as ant colony, evolutionary algorithms, iterated local search, simulated annealing, and tabu search. A survey of metaheuristics can be found in [BR03, Glo03]. Out of the many descriptions of metaheuristics that can be found in the literature, the following fundamental features can be highlighted:

- They are general strategies or templates that guide the search process.
- Their goal is to provide an efficient exploration of the search space to find (near) optimal solutions.
- They are not exact algorithms and their behavior is generally non deterministic (stochastic).
- They may incorporate mechanisms to avoid visiting non promising (or already visited) regions of the search space.
- Their basic scheme has a predefined structure.
- They may use specific problem knowledge for the problem at hand, by using some specific heuristic controlled by the high level strategy.

In other words, a metaheuristic is a general template for a non deterministic process that has to be filled with specific data from the problem to be solved (solution representation, specific operators to manipulate them, etc.), and that can tackle problems with high dimensional search spaces. In these techniques, the success depends on the correct balance between *diversification* and *intensification*. The term diversification refers to the evaluation of solutions in distant regions of the search space (with some distance function previously defined for the solution space); it is also known as *exploration* of the search space. The term intensification refers to the evaluation of solutions in small bounded regions, or within a neighbourhood (*exploitation* of the search space). The balance between these two opposed aspects is of the utmost importance, since the algorithm has to quickly find the most promising regions (exploration), but also those promising regions have to be thoroughly searched (exploitation).

We can distinguish two kinds of search strategy in metaheuristics. First, there are "intelligent" extensions of local search methods (trajectory-based metaheuristics in Figure 2.1). These techniques add some mechanism to escape from local optima to the basic local search method (which would otherwise stick to it). Tabu search (TS) [Glo86], iterated local search (ILS) [Glo03], variable neighbourhood search (VNS) [MH97] or simulated annealing (SA) [KGV83] are some techniques of this kind. These metaheuristics operate with a single solution at a time, and one (or more) neighbourhood structures. A different strategy is followed in ant colony optimization (ACO) [Dor92], particle swarm optimization (PSO) [Cle10] or evolutionary algorithms (EA) [Glo03]. These techniques operate with a set of solutions at any time (called colony, swarm or population, depending on the case), and use a learning factor as they, implicitly or explicitly, try to grasp the correlation between design variables in order to identify the regions of the search space with high-quality solutions (population-based techniques in Figure 2.1). In this sense, these methods perform a biased sampling of the search space.

A formal definition of metaheuristics can be found in [Luq06], with an extension in [Chi07]. The complete formulation with the concepts of state, dynamics, and execution of a metaheuristic are presented in the following definitions.

Definition 2.1.2 (Metaheuristic)**.** *A metaheuristic $\mathcal{M}$ is a tuple consisting of eight components as follows:*

$$\mathcal{M} = \langle \mathcal{T}, \Xi, \mu, \lambda, \Phi, \sigma, \mathcal{U}, \tau \rangle \ , \tag{2.4}$$

where:

- *$\mathcal{T}$ is the set of elements operated by the metaheuristic. This set contains the search space, and in many cases they both coincide.*
- *$\Xi = \{(\xi_1, D_1), (\xi_2, D_2), \ldots, (\xi_v, D_v)\}$ is a collection of v pairs. Each pair is formed by a state variable of the metaheuristic and the domain of said variable.*
- *μ is the number of solutions operated by $\mathcal{M}$ in a single step.*
- *λ is the number of new solutions generated in every iteration of $\mathcal{M}$.*
- *$\Phi : \mathcal{T}^\mu \times \prod_{i=1}^{v} D_i \times \mathcal{T}^\lambda \to [0,1]$ represents the operator that produces new solutions from the existing ones. The function must verify for all $x \in \mathcal{T}^\mu$ and for all $t \in \prod_{i=1}^{v} D_i$,*

$$\sum_{y \in \mathcal{T}^\lambda} \Phi(x,t,y) = 1 \ . \tag{2.5}$$

- *$\sigma : \mathcal{T}^\mu \times \mathcal{T}^\lambda \times \prod_{i=1}^{v} D_i \times \mathcal{T}^\mu \to [0,1]$ is a function that selects the solutions that will be manipulated in the next iteration of $\mathcal{M}$. This function must verify for all $x \in \mathcal{T}^\mu$, $z \in \mathcal{T}^\lambda$ and $t \in \prod_{i=1}^{v} D_i$,*

$$\sum_{y \in \mathcal{T}^\mu} \sigma(x,z,t,y) = 1 \ , \tag{2.6}$$

$$\forall y \in \mathcal{T}^\mu, \sigma(x,z,t,y) = 0 \vee \sigma(x,z,t,y) > 0 \wedge \tag{2.7}$$
$$(\forall i \in \{1,\ldots,\mu\}, (\exists j \in \{1,\ldots,\mu\}, y_i = x_j) \vee (\exists j \in \{1,\ldots,\lambda\}, y_i = z_j)) \ .$$

- *$\mathcal{U} : \mathcal{T}^\mu \times \mathcal{T}^\lambda \times \prod_{i=1}^{v} D_i \times \prod_{i=1}^{v} D_i \to [0,1]$ represents the updating process for the state variables of the metaheuristic. This function must verify for all $x \in \mathcal{T}^\mu$, $z \in \mathcal{T}^\lambda$ and $t \in \prod_{i=1}^{v} D_i$,*

$$\sum_{u \in \prod_{i=1}^{v} D_i} \mathcal{U}(x,z,t,u) = 1 \ . \tag{2.8}$$

- *$\tau : \mathcal{T}^\mu \times \prod_{i=1}^{v} D_i \to \{false, true\}$ is a function that decides the termination of the algorithm.*

The previous definition represents the typical stochastic behavior of most metaheuristics. In fact, the functions Φ, σ and $\mathcal{U}$ should be considered as conditional probabilities. For instance, the value of $\Phi(x,t,y)$ is the probability to generate the offspring vector $y \in \mathcal{T}^\lambda$, since the current set of individuals in the metaheuristic is $x \in \mathcal{T}^\mu$, and its internal state is given by the state variables $t \in \prod_{i=1}^{v} D_i$. One can notice that the constraints imposed over the functions Φ, σ and $\mathcal{U}$ enable them to be considered as functions that return the conditional probabilities.

2.1.1 Classification of Metaheuristics

There are many ways to classify metaheuristics [BR03]. Depending on the chosen features we can obtain different taxonomies: nature inspired vs. non nature inspired, memory based vs. memoryless, one or several neighbourhood structures, etc. One of the most popular classifications distinguishes *trajectory based* metaheuristics from *population based* ones. Those of the first type handle a single element of the search space at a time, while those of the latter work on a set of elements (the population). This taxonomy is graphically represented in Figure 2.2, where the most representative techniques are also included. A subgroup in population based metaheuristics can be labeled as Swarm Intelligence approaches, which are characterized to perform population dynamics inspired on collective behaviors of animals in Nature (bees, ants, birds, etc.). The next two sections describe these kinds of metaheuristic.

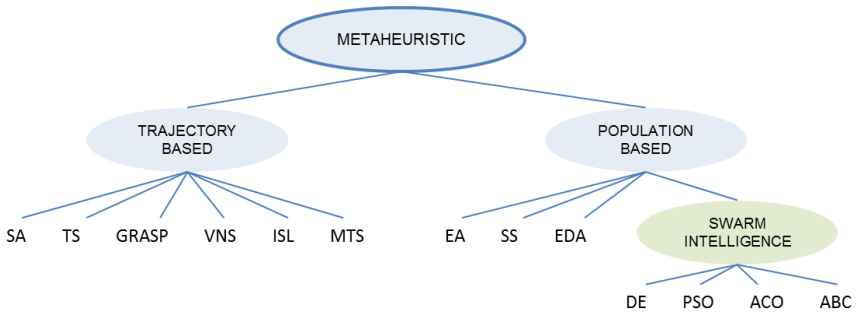


Figure 2.2: Classification of metaheuristics

Trajectory Based Metaheuristics

This section serves as a brief introduction to trajectory based metaheuristics. The main feature of these methods is the fact that they start from a single solution, and by successive neighbourhood explorations, update that solution, describing a trajectory through the search space. Most of this algorithms are extensions of the simple local search, which incorporates some additional mechanism for escaping local optima. This results in a more complex stopping condition than the simple detection of a local optimum. Widely used stopping criteria are completing some predefined number of iterations, finding some acceptable solution, or reaching some stagnation point.

• Simulated Annealing (SA)

Simulated Annealing (SA) was introduced in [KGV83], and is one of the oldest techniques among metaheuristics, and the first algorithm with an explicit strategy for escaping local optima. Its origins can be found in a statistical mechanism, called *metropolis*. The main idea in SA is to simulate the annealing process of a metal or glass. To avoid getting stuck in a local optimum, the algorithm always allows the selection of a solution with worse *fitness* value than the current one with some probability. The mechanism works as follows: in each iteration a solution s' is extracted from the neighbourhood $N(s)$ of current solution s; if s' has better fitness value than s, then s is discarded and s' is kept instead, otherwise s is replaced by s' only with a given probability that depends on a dynamic parameter T called temperature, and the difference between the fitness values of the two solutions, $f(s') - f(s)$.

• **Tabu Search (TS)**

Tabu Search (TS) is one of the metaheuristics that has been most successfully used to solve combinatorial optimization problems. The basics of this method were introduced in [Glo86]. The main idea in TS is the use of an explicit search history (short term memory), that serves both for escaping from local optima and for enhancing the diversity of the search process. This short term memory is called the tabu list, and keeps record of the last visited solutions, preventing the algorithm from visiting them again. At the end of each iteration, the best solution among the allowed ones is included in the list.

From the perspective of the implementation, keeping a list of full solutions is inefficient due to wasted memory consumption. Therefore, a commonly adopted alternative is to register the *movements* performed by the algorithm instead. In any case, the elements in the list can be used to filter the neighbourhood, producing a reduced set of eligible solutions named $N_a(s)$. Storing movements instead of complete solutions is more efficient, but causes a loss of information as well. In order to avoid this problem, an aspiration criterion is defined that permits the inclusion of a solution in $N_a(s)$ despite that solution being in the tabu list. The most widely used aspiration criterion is to permit solutions with better fitness values than the best fitness found so far.

• **GRASP**

Greedy Randomized Adaptive Search Procedure (GRASP) [FR95] is a simple metaheuristic that combines constructive heuristics with local search. GRASP is an iterative procedure with two phases: first, a solution is constructed; second, the solution undergoes an improvement process. The improved solution is the final result of the search process. A randomized heuristic is used for the construction of the solution in the first phase. Step by step, different components c are added to the partial solution s^p, initially empty. Each added component is randomly selected from a restricted list of candidates (RCL). This list is a subset of $N(s^p)$, the set of permitted components for the partial solution s^p. The components of the solution in $N(s^p)$ are sorted according to some problem dependent function η in order to generate the list.

The RCL contains the α best components in the set. In the extreme case of $\alpha = 1$, only the best component found is added to the list, thus resulting in a greedy construction method. In the other extreme, $\alpha = |N(s^p)|$, the component is chosen in a totally random way among all available components. Hence, α is a key parameter that determines how the search space is going to be sampled.

The second phase of the algorithm consists in a local search method to improve the previously generated solutions. A simple local search heuristic can be employed, or some more complex technique like SA or TS.

• **Variable neighbourhood Search (VNS)**

Variable neighbourhood Search (VNS) is a metaheuristic proposed in [MH97], that uses an explicit strategy to switch among different neighbourhood structures during the search. It is a very generic algorithm with many degrees of freedom to design variations or particular instances.

The first step is to define the set of neighbourhood descriptions. There are many ways this can be done: from random selection up to complex mathematical equations deduced using problem knowledge. Each iteration contains three phases: selection of a candidate, improvement phase, and finally, the movement. During the first phase, a neighbour s' is randomly chosen in the k^{th} neighbourhood of s. This solution s' acts then as the starting point for the second phase. Once

the improvement process is over, the resulting solution s'' is compared with the original, s. If s'' is better then it becomes the current solution and the neighbourhood counter is reset ($k \leftarrow 1$); if it is not better, then the process is repeated for the next neighbourhood structure ($k \leftarrow k+1$). The local search can be considered as the intensity factor, whereas the switches among neighborhoods can be considered as the diversity factor.

• Iterated Local Search (ILS)

The *Iterated Local Search* (ILS) metaheuristic [Glo03] is based on a simple yet effective concept. At each iteration, the current solution is perturbed and, to this new solution, a local search method is applied, to improve it. An acceptance test is applied to the local optimum obtained from the local search to determine whether it will be accepted or not. The perturbation method has an obvious importance: if it is not disruptive enough, the algorithm may still be unable to escape the local optimum; on the other side, if it is too disruptive, it can act as a random restarting mechanism. Therefore, the perturbation method should generate a new solution that serves as the starting point for the local search, but not so far away from the current solution as to be a random solution. The acceptance criterion acts as a balance method, since it filters new solutions to decide which can be accepted depending on the search history and the characteristics of the local optimum.

• Multiple Trajectory Search (MTS)

The *Multiple Trajectory Search* (MTS) is a modern trajectory based metaheuristic initially designed for multi-objective optimization showing accurate results in this domain [TC09]. However, it is currently becoming popular for large scale continuous optimization [TC08], since it showed the best performance in the special session on Large Scale Global Optimization of CEC'08 [TYS+07].

In MTS, after an initialization phase using simulated orthogonal array (SOA), a number of three different local search procedures are applied to each individual and selects the best from the three new solutions. *Local Search 1* searches along one dimension from the first dimension to the last dimension. *Local Search 2* is similar to *Local Search 1* except that it searches along about one-fourth of dimensions. In both local search methods, the search range (SR) will be cut to one-half until it is less than certain *0-threshold* (usually $1.0E^{-15}$) if the previous local search does not make improvement. In *Local Search 1*, on the dimension concerning the search, the solution's coordinate of this dimension is first subtracted by SR to see if the objective function value is improved. If it is, the search proceeds to consider the next dimension. If it is not, the solution is restored and then the solution's coordinate of this dimension is added by $0.5 \cdot SR$, again to see if the objective function value is improved. If it is, the search proceeds to consider the next dimension. If it is not, the solution is restored and the search proceeds to consider the next dimension. *Local Search 3* considers three small movements along each dimension and heuristically determines the movement of the solution along each dimension. In *Local Search 3*, although the search is along each dimension from the first dimension to the last dimension, the evaluation of the objective function value is done after searching all the dimensions, and the solution will be moved to the new position only if the objective function has been improved at this current evaluation.

Population Based Metaheuristics

Population based methods are characterized by working with a set of solutions at a time, usually named *the population*, unlike trajectory based methods, which handle a single solution. We here

Table 2.1: Different Types of EAs

Name	Representation	Details
Evolutionary Programming (EP)	Real values	Mutation and $(\mu + \lambda)$ Selection
Evolution Strategies (ES)	Real values and strategy parameters	Mutation, Recombination, and $(\mu + \lambda)$ or (μ, λ) Selection
Genetic Algorithms (GA)	Binary and Real values	Mutation, Recombination, and Selection
Genetic Programming (GP)	A computer program	Mutation, Recombination, and Selection

make a brief survey to some relevant techniques for this thesis; the interested reader can get more information in [Glo03] and [BR03].

• Evolutionary Algorithms (EA)

Evolutionary Algorithms (EAs) are loosely inspired on the theory of the natural evolution of the species. The techniques in this wide family follow an iterative stochastic process that operates a population (set) of tentative solutions, referred to as *individual* within this context, attempting to generate new tentative solutions with higher and higher fitness.

The general template of an EA has three phases, named after their natural equivalents: *selection*, *reproduction* and *replacement*. The whole process is repeated until some stopping criterion is met (generally, after a certain number of operations has been performed). The selection phase selects the fittest individuals from the current population, to be recombined later during the reproduction phase. Different kinds of selection strategies exist: *roulette wheel*, *tournament*, *random*, $(\mu+\lambda)$, or (μ, λ), where μ represents the number of parent solutions and λ the number of generated children. The resulting individuals from the recombination are modified by a mutation operator. Thus, the reproduction phase can be divided into two sub-phases: recombination and mutation. This are the usual operations in genetic algorithms, the rest of EAs having other so called "variation operators" like local search or ad-hoc techniques. Finally, the new population is formed with individuals from the current one, and/or the best newly generated individuals (according to their fitness or suitability values). This new population is used as the current population in the next iteration of the algorithm.

Depending on the individual representation and on how these phases are implemented, different instances of EAs arise. Table 2.1 summarizes the three major instantiation of EAs, namely *Evolutionary Programming* (EP), *Evolution Strategy* (ES), *Genetic Algorithm* (GA), and Genetic Programming (GP), and highlights their major differences.

• Estimation of Distribution Algorithms (EDA)

Estimation of Distribution Algorithms (EDAs) [LLIB06] have a similar behavior with respect to the previously presented EAs, and many authors even consider EDAs as a special kind of EA. Like EAs, EDAs operate on a population of candidate solutions, but, unlike them, do not use recombination and mutation to generate the new solutions, but a probability distribution mechanism instead.

Graphic probabilistic models are commonly used tools to represent in an efficient manner the probability distributions when working with EDAs. Bayesian networks are usually to represent the probability distributions in discrete domains, while Gaussian networks are most often applied for continuous domains.

• **Scatter Search (SS)**

Scatter Search (SS) is another metaheuristic whose basic principles were presented in [Glo98], and is currently receiving an increasing deal of attention from the research community. The algorithm's fundamental idea is to keep a relatively small set of candidate solutions (called the reference set, or *RefSet* for short), characterized by hosting diverse (distant in the search space) high-quality solutions. Five components are required for the complete definition of SS: initial population creation method, reference set generation method, subsets of solutions generation method, solution combination method, and improvement method. This algorithm is explicitly incorporating the exploration/exploitation idea in concrete steps and operations of its improvement loop.

• **Ant Colony Optimization (ACO)**

Ant Colony Optimization (ACO) [Dor92] is a swarm intelligent algorithm inspired by the foraging behavior of real ants in the search for food. This behavior can be described as follows: initially, ants explore the surrounding area of their nest or colony in a random fashion. As soon as an ant finds a food source, it starts carrying that food to the nest; as it does this, the ant continuously deposits a chemical known as pheromone in its path. The pheromone can be detected by other ants, thus guiding them to the food. This indirect communication among ants also serves to find the shortest path between the nest and the food.

ACO methods simulate this behavior to solve optimization problems. These techniques have two main phases: construction of a solution following a single ant's behavior, and updating of the artificial pheromone trail. There is no *a priori* synchronization between the phases, which can even be done simultaneously. These are powerful algorithms especially for searching in problems whose solutions are modeled as paths in a graph.

• **Artificial Bee Colony (ABC)**

The *Artificial Bee Colony* (ABC) [KB07] is a swarm intelligent metaheuristic, in which the position of a food source represents a possible solution to the optimization problem and the nectar amount of a food source corresponds to the quality (fitness) of the associated solution.

In ABC, at the first step, a randomly distributed initial population (food source positions) is generated. After initialization, the population is subjected to repeat the cycles of the search processes of the employed, onlooker, and scout bees, respectively. An employed bee produces a modification on the source position in her memory and discovers a new food source position. Provided that the nectar amount of the new one is higher than that of the previous source, the bee memorizes the new source position and forgets the old one. Otherwise, she keeps the position of the one in her memory. After all employed bees complete the search process, they share the position information of the sources with the onlookers on the dance area. Each onlooker evaluates the nectar information taken from all employed bees and then chooses a food source depending on the nectar amounts of sources.

2.1.2 Extended Models

In this section we discuss some techniques to allow the basic metaheuristic algorithms solve complex real problems; this is much needed, since the basic sheets for search is very general for them, and some lines of improvement are needed to actually face efficient resolution of complex problems.

The techniques we will mention here are relevant to our work; in particular we will describe some basics on hybridization, parallelization, and multi-objective resolution.

• Hybrid Metaheuristics

In recent years it has become evident that a skilled combination of a metaheuristic with other optimization techniques, a so called hybrid metaheuristic [Dav01], can provide a more efficient behavior and a higher flexibility. This is because hybrid metaheuristics combine their advantages with the complementary strengths of, for example, more classical optimization techniques such as branch and bound or dynamic programming.

A possible characterization of hybrid metaheuristic given in [BR03] is as follows: *a technique that results from the combination of a metaheuristic with other technique/s for optimization*, such as: metaheuristics, problem-specific algorithms or simulators, artificial intelligent or operational research techniques, and even, human interactions. There are several classifications of hybrid metaheuristics [DS03, BR03, NCM11, Rai06] focusing on different design issues: level of coupling techniques (weak, strong), control strategy (collaborative, integrative), and order of execution (sequential, interleaved, and parallel).

When hybridizing metaheuristics, the most common goal is to provide new algorithms with both, diversification and intensification abilities. This is usually approached by incorporating trajectory search methods into population based techniques. A representative kind of these techniques are the so called Memetic Algorithms [NCM11], consisting on the application of local search to individuals in evolutionary algorithms. Other approaches consider the *two-stage* (or *n-stage*) hybridization, in which a metaheuristic operates in a first stage of execution, and giving the control of the optimization procedure to other different technique in the second stage.

• Parallel and Distributed Metaheuristics

Even though the use of metaheuristics alone can significantly reduce the complexity and time length of the search process, that time can still remain too large for some real problems. With the development of cheap efficient platforms for parallel computation, it comes as natural to leverage on their power to accelerate the resolution process for these complex problems. There is an extensive literature on parallelization of metaheuristic techniques [AT02, Alb05, CMRR02] since it constitutes an interesting approach, not only for reducing computation times, but also for obtaining higher accuracy of the solution process (i.e., solutions of higher quality). This improvement is due to a new search model that enables a finer tuning between intensity and diversity.

When handling populations, parallelism comes out in a natural way, as different individuals may be operated independently. Hence, the performance of population based algorithms tends to improve as they are executed in parallel. From a high level viewpoint, parallel strategies for this kind of methods can be classified into two categories: (1) parallel computation, where the individual operations are performed in parallel, and (2) parallel population, where the algorithm's population is structured into smaller subpopulations.

One of the most frequently used models that follows the first strategy is the so called *master-slave* model (also known as global parallelization). Within this model, the central "master" process performs the population-scale operations (such as the selection method of an EA), while the slaves perform the independent individual-scale operations (such as the individual fitness value computation, mutation, and sometimes the recombination as well). This kind of strategy is mostly used in scenarios where the fitness value computation is a costly process (in computation time). Another popular strategy consists in accelerating the computation time by performing multiple independent

executions at a time (with no interaction among them) in many computers; upon completion of all the executions, the best solution found among all is kept. Again, this process does not change the global behavior of the algorithm, but reduces the computation wall clock time.

Besides the master-slave model, most parallel population-based algorithms found in the literature use some kind of structure for their population of individuals. This kind of model is specially used with EAs. Among the most popular models for structured populations are the *distributed model* or coarse grained, and the *cellular model* or fine grained [AT02].

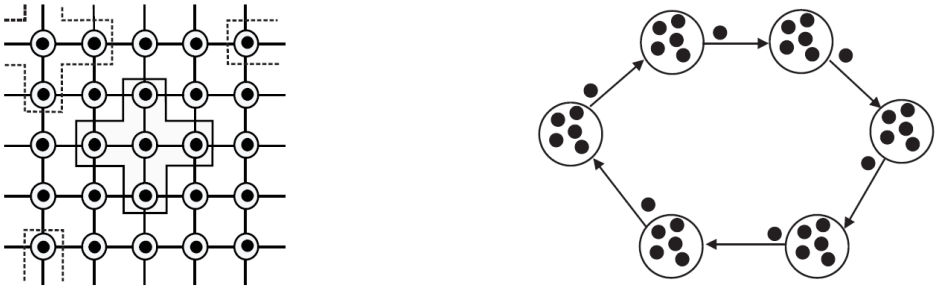


Figure 2.3: Structured population models: (left) cellular and (right) distributed

In the case of distributed algorithms [Alb05] (Figure 2.3 (right)), the population is divided into a set of smaller subpopulations or islands, each of which is then handled in parallel by a sequential metaheuristic. Islands cooperate by exchanging information (typically individuals); this cooperation is used to introduce new diversity into the subpopulations, keeping them from stagnating around local optima. The parameters required for the complete definition of this model include: the topology, which determines the directions of the logical communication channels among islands; the migration schedule, which determines at which moments of the execution the information exchanges will take place (since the communications are typically periodic, this parameter is normally reduced to the value of the migration period); the migration ratio, which determines the amount of information (i.e., number of individuals) exchanged; the selection and replacement criteria, which determine, in the case of migrating individuals, which individuals enter and leave each island. Finally, the communication among islands can be made to be synchronous, or asynchronous.

Alternatively, cellular metaheuristics [AD08] (Figure 2.3 (left)) are based on the concept of neighbourhood[1]. Each individual has a set of close individuals or *neighbours* according to some virtual superimposed regular structure (like in a crystal or a beehive) with which the exploitation of solutions will be performed. Exploration and diffusion of solutions to the rest of the population happens in a smooth fashion, due to the continuous overlap existing among the different neighborhoods, which lets high quality solutions to propagate over the population. This in turn could be done purely in sequential or in parallel (either in multiple CPUs or inside a GPU).

Besides these basic models, there are many existing hybrid models in the literature that combine two-tiered strategies. For instance, a commonly found strategy is one in which coarse grain is used in the higher tier, and a cellular model is used within each subpopulation.

[1]Once again, the concept of *neighbourhood* for a cellular metaheuristic is different from the ones previously mentioned for different contexts, such as trajectory based methods.

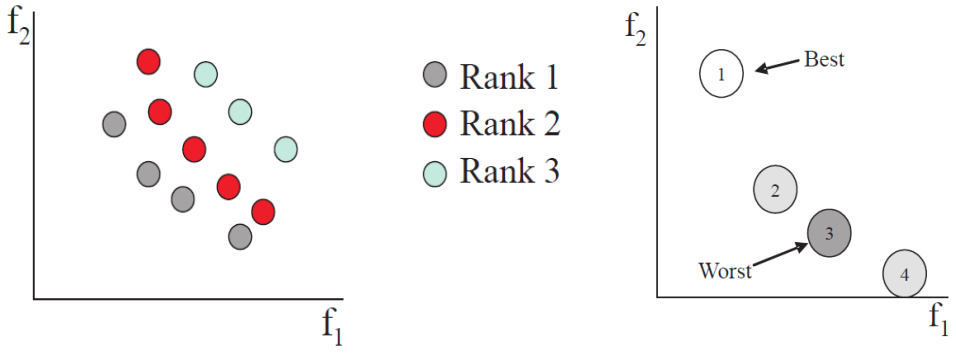


Figure 2.4: Example of sorting (*ranking*) of solutions in a bi-objective MOP (left). Qualifying non-dominated solutions according to the density of solutions in a bi-objective MOP (right)

• Multi-objective Metaheuristics

Up to now, we have dealt with mono-objective problems, since we want the algorithm to compute one single global optimum (even if the algorithm is using for this other fit solutions also). But in practice, lots of problems need to address the resolution of a task where two or more objective functions are to be optimized, being they all equally important for the task. The use of Pareto optimality based techniques means dealing with a set of non-dominated solutions, which requires some specific mechanisms to handle them. Additionally, this set must be diverse enough to cover the whole front. Although depending on the algorithm, there are many different issues to cope with, the following ones are commonly found in many of the existing techniques: fitness function, diversity management, and constraint handling mechanisms.

In the life-cycle of any metaheuristic technique there always exists a phase in which all the solutions must be sorted to pick one (or more) of them. Examples of these phases are the selection and replacement mechanisms in EAs, or the reference set updating procedure in scatter search algorithms. In single-objective optimization, the fitness is a single (scalar) value, and thus, the sorting is done according to it. However, in the multi-objective domain, the fitness consists of a vector of values (one value per objective function), and as a consequence, the sorting is not straightforward.

The dominance relationship is the key issue in Pareto optimality based techniques, since it allows us to sort all the solutions. Actually, this relation defines a partial order relationship. Different methods have been proposed in the literature [CCLV07, Deb01], which basically transform the fitness vector into a single value. Actually, this kind of strategy was first proposed by Goldberg in [Gol89] for guiding a GA population towards the Pareto front of a given Multi-Objective problem (MOP). The basic idea behind it consists in successively finding solutions that are non dominated by other solutions (the best ones according to the dominance relationship). Those solutions are referred to as non-dominated. The highest possible value is assigned to those solutions. Then, the next fitness value is assigned to the non-dominated solutions remaining after the previous ones are removed from the population. The procedure continues until there is no solution left in the population. Figure 2.4 (left) depicts an example of the behavior of this sorting mechanism (where f_1 and f_2 are the objective functions which should be minimized). This strategy is known as *ranking*.

Even though the Pareto dominance based fitness function guides the search towards the Pareto front, this approximation is not enough when a MOP is tackled, and also diversity in the front is useful to the decision maker. Although different approximations exist in the literature [CCLV07], many of the state-of-the-art ones are based on complementing the dominance based fitness function with a density estimator, which measures the crowd around a solution inside the objective space. Thus, given two solutions with the same fitness function value (*ranking* and *strength*), the density estimator discriminates between them attending to their diversity. Let us consider the set of solutions in Figure 2.4 (right). In this figure, solution 1 can be considered as the best one regarding the density of solutions since it resides in the less crowded area of the front. On the other hand, solution 3 is the worst one, since it is surrounded by many other close solutions. There exist some well-known density estimators in the literature [CCLV07]: niching, adaptive grid, crowding, and the k-nearest neighbour distance.

Concerning constraints handling mechanism, the scheme used by most of the state-of-the-art metaheuristics for multi-objective optimization consists in considering that feasible solutions (those which do not violate any constraint) are better than non-feasible ones, regardless of their objective values [Deb00, Deb01]. Thus, given two solutions there are three possible cases:

1. If both solutions are feasible, the fitness function explained before should be used to distinguish between them; in case of being non-dominated, a density estimator must be applied.
2. If only one of them is a feasible solution, it should be considered as the best one.
3. If both solutions are infeasible, the one which less violates the constraints is considered to be the best.

The influence of solving multi-objective problems in the expected interest on a technique is so important in research today that we include this topic in the thesis to avoid making a smaller contribution just considering mono-objective problems. This needs a considerable effort in design and analysis of algorithms, as well as in entering this field full of especial topics, but definitely a must in a modern thesis with a vocation of impact in the present hard panorama of research.

2.1.3 Statistical Validation Procedure

As previously explained, metaheuristics are stochastic based algorithms with different random components in their operations. Opposite to deterministic procedures, for which, just a single execution is required, when working with metaheuristics, performing a series of independent runs for each algorithm's configuration is a mandatory task in order to obtain a distribution of results. In this case, it is possible to compute a global indicator (median, mean, standard deviation, etc.) from the resulted distribution to measure the performance of the studied algorithm. Nevertheless, using one single global indicator to directly compare metaheuristics should lead empirical analyses to biased conclusions. Therefore, the correct practice is to compare the distributions of results by means of statistical tests, which are indispensable tools to validate and to provide confidence to our empirical analysis.

The standard procedure, recommended by the scientific community [She07, GMLH09], for the statistical comparison of metaheuristics lies in the use of *parametric* or *non-parametric* tests. Parametric tests show a high precision to detect differences in comparisons, although they are restricted to distributions fulfilling three specific conditions: *independency*, distributions are obtained from independent executions; *normality*, they follow a Gaussian distribution; and *heteroskedasticity*, indicating the existence of a violation of the hypothesis of equality of variances. Non-parametric

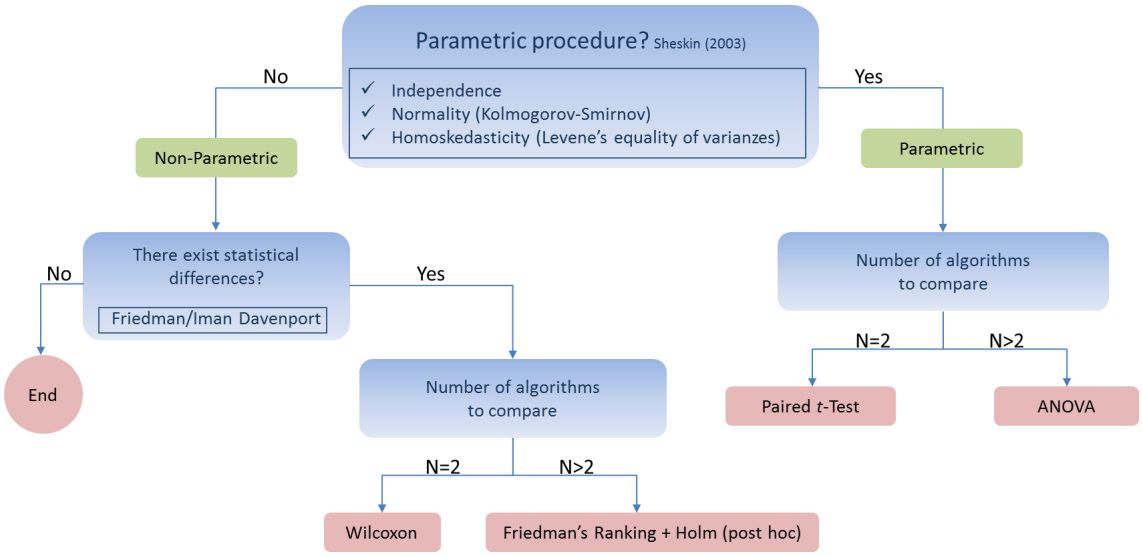


Figure 2.5: Statistical validation procedure for experimental results

tests also show a successful performance, although they are less restrictive, since they can be applied regardless of the three previous conditions. Among all these tests, we can find procedures to perform rankings, pair-wise comparisons, and multiple post hoc comparisons.

In this thesis, we have adopted a formal statistical procedure, as recommended in [She07, GMLH09], to validate our results and to compare our proposals with other techniques in the current state of the art. Our null hypothesis (equality of distributions) has been set with a confidence level of 95%, meaning that statistical differences can be found in distributions when resulted tests are with a $p\text{-}value < 0.05$.

Figure 2.5 illustrates this statistical procedure. The first step consists on choosing the kind of tests to use: parametric or non-parametric. For this, as empirical executions are always independently done in our experiments, we directly perform a *Kolmogorov-Smirnov* test to check the samples are distributed according to a normal distribution (Gaussian) or not. After this, the homoskedasticity (i.e., equality of variances) is then checked using the *Levene* test. If all distributions are normal and the Levene returns a positive value, then we use the parametric procedure. Otherwise, we use the non-parametric one. In the case of parametric procedure, a *Paired t-test* or an *ANOVA* test are performed depending on the number of distributions that we are comparing: 2 or more than 2, respectively. For non-parametric, *Friedman's* or *Iman Davenport's* tests are first performed in order to check whether statistical differences exist or not. If yes, a *Wilcoxon's* or *Holm's* tests are performed depending on the number of distributions to compare: 2 or more than 2, respectively. Otherwise, the procedure finishes without rejecting the null hypothesis.

Chapter 3

Fundamentals of Particle Swarm Optimization (PSO)

3.1 PSO: Introduction

Particle swarm arose from a series of experimental simulations performed by Reynolds in 1987 [Rey87] and continued by Heppner and Grenander in 1990 [HG90] in which, the dynamics of social agent systems inspired on bird flocks were studied. In these simulations, a point on the screen was defined as food, called the "cornfield vector"; the idea was for birds to find food through social learning, by observing the behavior of nearby birds, who seemed near the food source. Kennedy and Eberhart (1995) [KE95] experimented with the optimization potential of particles' dynamics and modified the algorithm to incorporate topological neighborhoods and multidimensional search. In this way, each particle belongs to a social neighborhood, and its social influence will result from the observation of its neighbors. It means that a particle will be affected by the best point found by any member of its topological neighborhood. The definition of a neighborhood topology is simply the characterization of a social network, represented as a graph, where each individual is represented as a vertex and an edge exists between two individuals if they can influence one another.

The main difference between the PSO paradigm and other instances of population based algorithms, such as EAs, is memory and social interaction among the individuals. In the other paradigms, the important information an individual possesses, usually called *genotype*, is its current position. In PSO, the really important asset is the previous best experiment. This is the drive toward better solutions: each individual stores the best position found so far. The mechanism responsible for sampling, the equivalent of recombination, is the imitative behavior of the individuals in the particle's social neighborhood. The fact that this behavior is stochastic in nature accounts for the fact the algorithm can sample new solutions in areas not yet explored by the swarm.

There exist numerous descriptions of PSO in the literature [KE01, Men04, PKB07, Cle10], with detailed formulations of its components and parameters. Our aim in this chapter is to introduce the PSO from a new "morphological" perspective. First, we describe the canonical PSO and successive standard versions (2006, 2007, and 2011) proposed by research community in this field [PCG11]. After this, we emphasize the most interesting innovations provided by representative versions of this algorithm. We also describe other related metaheuristics that have been partially used in this thesis. Finally, we perform a general review of the impact of PSO publications in the literature concerning benchmarking studies and applications.

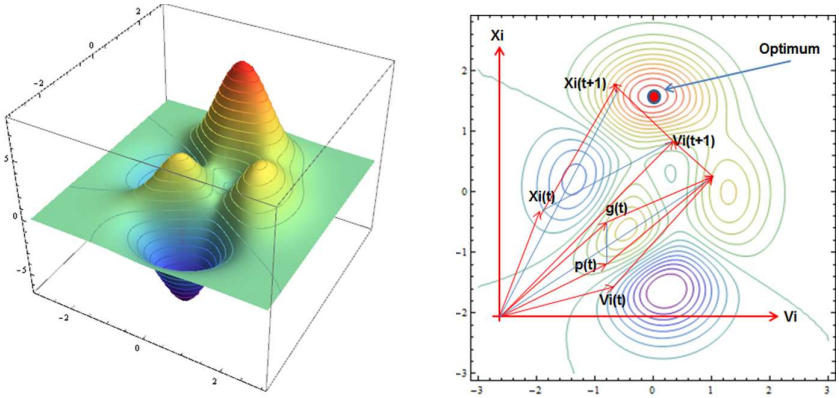


Figure 3.1: Particle swarm optimization vector dynamics. A new position of particle $\mathbf{x}_i^{t+1}$ is reached after velocity calculation

3.1.1 Canonical PSO

Canonical particle swarm optimization [KE01] works by iteratively generating new particles' positions located in a given problem search space. The position $\mathbf{x}_i$ represents a set of Cartesian coordinates describing a point in solution search space. Each one of these new particles' positions are calculated using the particle's current position, the particle's previous velocity, and two main informant terms: the particle's best previous location, and the best previous location of any of its neighbors.

Formally, in canonical PSO each particle's position vector $\mathbf{x}_i$ is updated each time step t by means of the Equation 3.1. From a graphic point of view, Figure 3.1 shows a representative illustration of a particle's position updating operation in a typical optimization problem landscape.

$$\mathbf{x}_i^{t+1} \leftarrow \mathbf{x}_i^t + \mathbf{v}_i^{t+1} \tag{3.1}$$

where $\mathbf{v}_i^{t+1}$ is the velocity vector of the particle given by

$$\mathbf{v}_i^{t+1} \leftarrow \omega \cdot \mathbf{v}_i^t + U^t[0, \varphi_1] \cdot (\mathbf{p}_i^t - \mathbf{x}_i^t) + U^t[0, \varphi_2] \cdot (\mathbf{b}_i^t - \mathbf{x}_i^t) \tag{3.2}$$

In this formula, $\mathbf{p}_i^t$ is the personal best position the particle i has ever stored, $\mathbf{b}_i^t$ is the position found by the member of its neighborhood that has had the best performance so far. Acceleration coefficients φ_1 and φ_2 control the relative effect of the personal and social best particles, and U^t is a diagonal matrix with elements distributed in the interval $[0, \varphi_i]$, uniformly at random. Finally, $\omega \in (0, 1)$ is called the inertia weight and influences the tradeoff between exploitation and exploration. High values of inertia promotes the exploration, although inducing fast oscillation around basin of attraction. A low inertia weight exploits a certain promising region and keep the particle from oscillating too fast without accurately exploring it. In addition, inertia avoids the use of a velocity constriction threshold $Vmax$ [KE01] that, in spite of preventing the explosion when velocity increases too much, it incorporates arbitrariness and problem dependence.

An equivalent version of the velocity equation was reported in [CK02], where Clerc's constriction coefficient χ is used instead of inertia weight as shown in Equation 3.3. This coefficient resulted after analyzing the trajectories of particles in the system that simply explodes after a few iterations, when the algorithm is run without restraining the velocity in some way.

$$\mathbf{v}_i^{t+1} \leftarrow \chi \cdot \left(\mathbf{v}_i^t + U^t[0,\varphi_1] \cdot (\mathbf{p}_i^t - \mathbf{x}_i^t) + U^t[0,\varphi_2] \cdot (\mathbf{b}_i^t - \mathbf{x}_i^t)\right) \tag{3.3}$$

$$\chi \leftarrow \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|},\ with\ \varphi \leftarrow \sum_i \varphi_i,\ and\ \varphi > 4 \tag{3.4}$$

Constriction coefficient χ is calculated, by means of Equation 3.4, from the two acceleration coefficients φ_1 and φ_2, being the sum of these two coefficients what determines the χ to use. Usually, $\varphi_1 = \varphi_2 = 2.05$, giving as results $\varphi = 4.1$, and $\chi = 0.7298$ [ES00, Tre03]. With these values the constriction method proposed by Clerc [CK02] results in convergence over time, and the amplitude of particles' oscillations decreases along with the optimization process.

The advantage of using constriction is that there is no need to use $Vmax$ nor to guess the values for any parameters governing convergence and preventing explosion. Subsequent experiments [ES00] concluded that it seems a good option to set $Vmax$ to $Xmax$, the dynamic range of each variable in each dimension. The result is a PSO with no problem-specific parameters, considered the canonical particle swarm algorithm.

Algorithm 1 Pseudocode of Canonical PSO

```
1: S ←initializeSwarm(Ss)
2: computeLeader(b)
3: while t < MAXIMUM_t do
4:    for each particle i in S do
5:       v_i^{t+1} ←updateVelocity(ω, v_i^t, x_i^t, φ1, p_i^t, φ2, b^t)   //Equations 3.2 or 3.3
6:       x_i^{t+1} ←updatePosition(x_i^t, v_i^{t+1})   //Equation 3.1
7:       evaluate(x_i^{t+1})
8:       p_i^{t+1} ←update(p_i^t)
9:    end for
10:   b_i^t ←updateLeader(b_i^t)
11: end while
```

As shown in Algorithm 1, the canonical PSO starts by initializing the swarm (Line 1), which includes both the positions and velocities of the particles. The corresponding p^i of each particle is randomly initialized, and the leader b is computed as the best particle of the swarm (Line 2). Then, for a maximum number of iterations, each particle *flies* through the search space updating its velocity and position (Lines 5 and 6), it is then evaluated (Line 7), and its personal best position p^i is also updated (Line 8). At the end of each iteration, the leader b is also updated.

3.1.2 Standard Versions of PSO

In related literature, there exist a number of works in which proposed approaches are compared against PSO versions so called "the standard one", although they are never the same and they never use homogeneous parameters. This motivated a group of researches in this field, supervised by James Kennedy and Maurice Clerc, to provide the academic community with a validated Standard PSO to be used in analysis and comparisons. This PSO version does not intend to be the best one, but simply a suggested approach very near to the original version (1995), with a series of improvements based on recent works. Therefore, in 2006 appeared the first standard of PSO which brought a few changes with regards to the canonical one, mostly on parameter setting. However, with the following standards, 2007 and 2011, several significant improvements were introduced concerning rotation invariance.

• **Standard PSO 2006**

The first standard PSO [PCG11] performs in a similar way to canonical one, although using an specific parameter setting as shown in Table 3.1. When a particle exceeds the limits of problem bounds, the position is set to the upper/lower value and the velocity is reset to zero. The neighborhood topology is random, meaning that the number of particles that informs a given one may be any value between 1 (for each particle informs itself) and Ss, the swarm size.

Table 3.1: Parameter settings in Standard PSO 2006

Parameter Value	Description
$Ss = 10 + 2 \cdot \sqrt{D}$	Swarm size, where D is the dimension of the search space
$K = 3$	Maximum number of particles informed by a given one
$T = (1 \cdots Ss)$	Topology, randomly (uniform) modified after each step if there has been no improvement
$\omega = \frac{1}{2 \cdot \ln(2)}$	Inertia, first cognitive/confidence coefficient
$\varphi_1 = \varphi_2 = 0.5 + \ln(2)$	Acceleration coefficients, second cognitive/confidence coefficients

Concerning the initialization of particles, initial positions are chosen at random inside the search space (which is supposed to be a hyperparallelepid, and even often a hypercube), according to a uniform distribution. Initial velocity vectors are calculated as the difference of two random positions. The resulting distribution is not even uniform, as for any method that uses a uniform distribution independently for each component. In this sense, authors argued that, in spite of this not being the best initialization method, it is close to the one of the original PSO.

• **Standard PSO 2007**

Standard PSO 2007 [PCG11] introduces a series of optional differences with regards to Standard 2006, although the improvement of these changes is not always clear and in fact their effect may only be marginal. The main differences can be enumerated as follows:

1. When a particle i has no better informant than itself, it implies that $\mathbf{p}_i = \mathbf{b}_i$ and Equation 3.2 (or 3.3) reduces to $\mathbf{v}_i^{t+1} = \omega \cdot \mathbf{v}_i^t + U^t[0, \varphi_1] \cdot (\mathbf{p}_i^t - \mathbf{x}_i^t)$.

2. Optional "*non sensitivity to rotation of the landscape*". When this option is activated, vectors $(\mathbf{p}_i^t - \mathbf{x}_i^t)$ and $(\mathbf{b}_i^t - \mathbf{x}_i^t)$ are replaced by rotated vectors. In this way, the final distribution of the next possible positions is not dependent on the system of coordinates.

3. Optional "*random permutation*" of particles before each iteration. As authors noticed, this operation is time consuming, but with no clear improvement.

4. Optional "*clamping position*". It refers to the decision to take when a particle is located out of the problem bounds. Clamping a particle sets its position to the problem bounds and the velocity to zero. No clamping (and no evaluation) may induce an infinite run if the stop condition is the maximum number of evaluations.

5. Probability Pr of a given particle being an informant of another one. In Standard PSO 2006, it was implicit by building the random neighborhood topology. In Standard 2007, the default value is directly computed as a function of (Ss, K), formally $Pr = 1 - (1 - \frac{1}{Ss})^K$. The topology is exactly the same as in Standard 2006, although in this standard it can be set at will by the researcher.

6. The search space can be *quantised* to transforms continuous variables to discrete ones. It consists of a *Mid-Thread uniform* quantiser method $Q(x) = \Delta \cdot \lfloor x/\Delta + 0.5 \rfloor$, with the quantum step set here to $\Delta = 1$.

The remaining parameters are set as specified in Table 3.1, and initialization of particles positions and velocities are performed as in the previous standard.

• Standard PSO 2011

In spite of the "non rotation invariance" affecting the PSO was already approached in Standard 2007 by rotating cognitive vectors, the performance of this previous version is still not satisfying. The authors argued the following reason: each iteration step, the set of all new possible positions is coordinate dependent. These new possible positions are a combination of two hypercubes whose sides are parallel to the axes, with uniform probability distribution inside each rectangle. The resulting combination is also a hypercube, but with a non uniform distribution (more dense near de center).

To solve this issue, an intuitive approach is to define a domain where new generated positions are not coordinate dependent. There are several possible shapes to define such a domain: hypershpere, hypercylinder, hypercone, etc. From these, preliminary experiments suggested that a hypersphere $HS(Gr, \parallel Gr - \mathbf{v}_i^t \parallel)$ is a good compromise, where Gr is the center of gravity with regards to cognitive factors $\mathbf{p}$ and $\mathbf{g}$. Therefore, the core of this Standard PSO 2011 [PCG11] consists on the velocity vector (v_{g+1}^i) calculation which is given by Equation 3.5.

$$v_i^{t+1} \leftarrow \omega \cdot v_i^t + Gr_i^t - x_i^t + HS(Gr, \parallel Gr - \mathbf{v}_i^t \parallel) \tag{3.5}$$

with

$$Gr_i^t \leftarrow \begin{cases} \frac{x_i^t + p_i'^t}{2} & \text{if } p_i'^t = l_i'^t \\ \frac{x_i^t + p_i'^t + l_i'^t}{3} & otherwise \end{cases} \tag{3.6}$$

$$p_i'^t \leftarrow x_i^t + c \cdot (p_i^t - x_i^t) \tag{3.7}$$

$$l_i'^t \leftarrow x_i^t + c \cdot (l_i^t - x_i^t) \tag{3.8}$$

In these formulas, p_i^t is the best solution that the particle i has seen so far, l_i^t is the best particle of a neighborhood of k other particles (also known as the *social best*) randomly (uniform) selected from the swarm. The acceleration coefficient $c > 1$ is a normal (Gaussian) random value with $\mu = 1/2$ and $\rho = 1/12$. This coefficient is sampled anew for each component of the velocity vector. Finally, HS is the distinctive element of the Standard PSO 2011 with regards to the previous ones. It is basically a random number generator within a Hypersphere space, with Gr as center of gravity. That is, Gr is calculated as the equidistant point to p'^t, l'^t, and x^t.

In addition to rotation invariance mechanism, several differences can be observed in Standard PSO 2011 with regards to the previous standard in 2007.

1. Normalization of the search space if it is possible, then being $[0, normalization]^D$. There exist problems for which the range of values is not the same for all variables. In such a case, using sphere domains may lead to bad performance.

2. Optional distribution of random factors: Gaussian distribution instead of uniform one. It has been experimented performing normal random generators better than uniform ones.

3. For each run, the swarm size is now randomly chosen around a number of 40 particles. The previous swarm size is given by a formula which is dimension dependant, often used to come out as far form the optimal.

4. The velocity initialization has been slightly modified in order to avoid particles to leave the search space at the very first steps.

The remaining parameters are set as specified in Table 3.1, and initialization of particles positions is performed as in the previous standards.

3.1.3 Prominent Versions of PSO

At the same time that canonical and standard PSO were defined, mostly in the last decade, a great number of different versions of this algorithm have appeared that incorporate new formulations and additional mechanisms with the twofold motivation of: enhancing its behavior (competitiveness), and adapting it to particular problem conditions (versatility). In this sense, exhaustive taxonomies [PKB07, SM09] in the literature have reported around a hundred of PSO versions, although they can be classified by similarity of attributes in few general categories. Almost every single PSO in the literature can be categorized by, at least, one of the following features: different solution encoding, hybridization, novel velocity formulation, topology neighborhood, and swarm structure. Moreover, some of these categories are related to each other, so that, a given topology can influence the velocity calculation, and the swarm structure can determine the complementary technique for hybridization. In spite of PSO versions being too numerous for us to describe every one of them, we have considered to characterize here a representative set of variants with quite different features and covering, as much as possible, the whole set of categories.

• Binary PSO

Kennedy and Eberhart proposed in [PCG11] an intuitive alteration of the canonical algorithm for binary solution encoding. In this version, the velocity is used as a probability threshold to determine whether $x_i(k)$, the k^{th} component of $\mathbf{x}_i$, should be evaluated as a zero or a one. For this, they used a "sigmoid" function $s(v_i(k))$ as in Equation 3.9.

$$s(v_i(k)) \leftarrow \frac{1}{1 + e^{-v_i(k)}} \tag{3.9}$$

Then, once generated a random number $r = U(0, 1)$ for each particle variable and compared it to $s(v_i(k))$, if r is less than the threshold, then $x_i(k)$ is interpreted as 1, otherwise as 0. In this variation, each velocity component is limited to $v_i(k) < Vmax$, where $Vmax$ is some value typically close to 6.0. This setting prevents the probability of the particle element assuming either a value of 0 or 1 from being too high. Though this discrete PSO has been shown to be capable of optimizing several combinatorial problem [KS98], it is limited to only discrete problems with binary-valued solution elements.

Several other binary versions of PSO can be found in the literature [Cle05, PFE05]. Nevertheless, all these versions consist of *ad hoc* adaptations from the original PSO, since notions of velocity or direction vectors have no natural extensions for bit-strings, so that their performance are still improvable.

• **Discrete PSO**

Concerning a PSO for arbitrary discrete alphabets, there exists no original proposal as in the binary case, but there is a series of discrete approaches for integer encoding and permutations.

An early but effective discrete particle swarm was proposed by Yoshida et al. [YKF+00], by means of which, the velocity update equation can be adapted for use with integer variables by discretizing the values before using them in the velocity update rule. In this approach, discrete variables can freely mixed with continuous ones, as long as the appropriate update equations require a different encoding.

More recently, Consoli et al. [CMPDDM10] proposed a new Discrete PSO without considering any velocity since, from the lack of continuity of the movement in a discrete space, the notion of velocity has a fuzzy meaning; however they kept the attraction of the best positions. They interpret the weights of the updating equation, as probabilities that, at each iteration, each particle has a random behavior, or acts in a way guided by the effect of an attraction. The moves in a discrete or combinatorial space are jumps from one solution to another. The attraction causes the given particle to move towards this attractor if it results in an improved solution.

A last attempt corresponds to Set-Based PSO [CZZ+10] for discrete optimization. This version uses a set-based representation scheme that enables it to characterize the discrete search space by defining the candidate solution as a crisp set and the velocity update rule as a set with probabilities. All arithmetic operators are then adapted to generate solutions in crisp sets domains. In this way, the authors propose a base method to PSO discretization that used in the case of Comprehensive Learning PSO (next described), as well as other basic versions.

• **Geometric PSO**

Geometric Particle Swarm Optimization (GPSO) [MCP07] enables us to generalize PSO to virtually any solution representation in a natural and straightforward way, extending the search to other search spaces, such as combinatorial ones. The key issue in GPSO consists in using a multi-parental recombination of particles (solutions) which leads to the generalization of a *mask-based crossover* operation, proving that it respects four requirements for being a *convex combination* in a certain space. A convex combination is an affine combination of vectors where all coefficients are non-negative. When vectors represent points in the space, the set of all convex combinations constitutes the convex hull. This way, the mask-based crossover operation substitutes the classical *movement* in PSO and *position update* operations, initially proposed for continuous spaces. This property has been demonstrated for the cases of Euclidean, Manhattan and Hamming [MCP07].

For the particular case of Hamming spaces, a *three-parent mask-based crossover* (3PMBCX) is defined as follows:

Definition 3.1.1. *Given three parents a, b and c in $\{0,1\}^n$, generate randomly a crossover mask of length n with symbols from the alphabet $\{a,b,c\}$. Build the offspring o by filling each position with the bit from the parent appearing in the crossover mask at the position.*■

In a convex combination, weights w_a, w_b and w_c indicate (for each position in the crossover mask) the probability of having the symbols a, b or c, respectively. The pseudocode of the GPSO algorithm for Hamming spaces is illustrated in Algorithm 2. For a given particle i, three parents take part in the 3PMBCX operator (line 9): the current position $\mathbf{x}^i$, the social best position $\mathbf{b}^i$ and the personal best position found $\mathbf{p}^i$ (of this particle).

The weight values ω_a, ω_b and ω_c indicate (for each element in the crossover mask) the probability of having values from the parents $\mathbf{x}_i$, $\mathbf{b}_i$ or $\mathbf{p}_i$, respectively. These values associated to each parent

Algorithm 2 Pseudocode of GPSO for Hamming Spaces

```
1: S ← initializeSwarm(Ss)
2: computeLeader(b)
3: while g < MAXIMUM_t do
4:    for each particle i in S do
5:       x_i^{t+1} ← 3PMBCX(x_i^t, ω_a, b_i^t, ω_b, p_i^t, ω_c)
6:       x_i^{t+1} ←mutate(x_i^{t+1}, p_μ)
7:       evaluate(x_i^{t+1})
8:       p_i^{t+1} ← update(b_i^t)
9:    end for
10:   b_i^{t+1} ← updateLeader(b_i^t)
11: end while
```

represent the *present* influence of the current position (ω_a), the *social* influence of the global best position (ω_b), and the *individual* influence of the historical best position found (w_c). A restriction of the geometric crossover forces ω_a, ω_b and ω_c to be non-negative and add up to one.

In summary, the GPSO operates as follows: in the first phase, uniform random initialization of particles is carried out by means of the $SwarmInitialization()$ function (Line 1). In a second phase, after the evaluation of particles (line 4), personal and social positions are updated (lines 5 and 6, respectively). Finally, particles are "moved" by means of the 3PMBCX operator (line 9). In addition, with a certain probability, a simple *bit-flip* mutation operator (line 10) is applied in order to introduce diversity in the swarm to avoid early convergence with a probability of p_μ. As evaluated in [AGNJT07], the *three-parent mask-based crossover* used in GPSO makes the offspring inherit the shared selected features present in the three parents involved in the mating.

• Bare Bones

Bare bones PSO was proposed by Kennedy in 2003 [Ken03] with the main feature that position and velocity update rules are substituted by a procedure that samples a parametric probability density function. In this first proposal, the particle's position are calculated by means of Gaussian distribution as in Equation 3.10. The idea was inspired by a plot distribution of positions attained by single canonical PSO moving in one dimension under the influence of fixed $\mathbf{p}^i$ and $\mathbf{b}^i$. The empirical distribution resembled a bell curve centered at μ^i.

$$\mathbf{x}_{t+1}^i \leftarrow N(\mu_t^i, \sigma_t^i), \text{ with } \mu_t^i \leftarrow \frac{\mathbf{p}_t^i + \mathbf{b}_t^i}{2} \text{ and } \sigma_t^i \leftarrow \mid \mathbf{p}_t^i - \mathbf{b}_t^i \mid \tag{3.10}$$

An improved version of bare bones was later proposed [RB06], in which the Gaussian distribution is replaced with a Lévy distribution. The Lévy distribution is bell-shape like the Gaussian one and is also stable distribution. It provides a tunable parameter α, which interpolates between the Cauchy ($\alpha = 1$) and Gaussian ($\alpha = 2$) distributions. This parameter can be used to control the width of the density curve. After an empirical analysis in [RB06], it was shown that for a value of $\alpha = 1.4$, the "Lévy Particle Swarm" reproduces the behavior of canonical PSO.

• Fully Informed Particle Swarm

As previously explained, if Clerc's constriction coefficient χ is used instead of inertia weight to formulate the velocity update, this coefficient is calculated by means of Equation 3.4, from the two acceleration coefficients φ_1 and φ_2, being the sum of these two coefficients what determines

the χ to use. Usually, $\varphi_1 = \varphi_2 = 2.05$, giving as results $\varphi = 4.1$, and $\chi = 0.7298$ [ES00, Tre03] which provides stability and proper convergence behavior to this algorithm. As stated by Mendes in [MKN04, MMWP05], this fact implies that the particle's velocity can be adjusted by any number of informant terms, as long as acceleration coefficients sum to an appropriate value, since important information given by other neighbors about the search space may be neglected through overemphasis on the single best neighbor. With this assumption, Mendes et all.(2004) [MKN04] proposed the Fully Informed PSO (FIPS), in which a particle uses information from all its topological neighbors.

In FIPS, the value φ, that is, the sum of the acceleration coefficients, is equally distributed among all the neighbors of a particle. Therefore, for a given particle i with position $\mathbf{x}_i$, φ is broken up in several smaller coefficients $\varphi_j = \varphi/|\mathcal{N}_i|, \forall j \in \mathcal{N}_i$. Then, the velocity is updated as follows:

$$\mathbf{v}_i^{t+1} \leftarrow \chi \left[\mathbf{v}_i^t + \sum_{j \in \mathcal{N}_i} U^t \left[0, \varphi_j\right] \cdot (\mathbf{p}_j^t - \mathbf{x}_i^t) \right], \tag{3.11}$$

where $\mathcal{N}_i$ is the set of neighbors of the particle i, and following the neighborhood a given topology. Figure 3.2 illustrates the topologies used by Mendes et al. [MKN04] as the ones with most successful performances in a previous work [KM02]. These topologies are: All, Ring, Square, Four-Clusters, and Pyramid. Their results show that the Square topology (with 4 informants) outperforms the other ones. Indeed, the fact of defining these neighborhoods in the swarm makes the particles to be influenced only by a certain number of neighbors, and connected with static links in the graph.

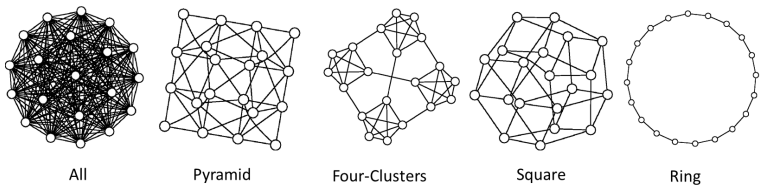


Figure 3.2: Topologies used by Mendes et al. [MKN04]. Each particle has a number of fixed neighbors in the swarm (All=N-1; Ring=2; Four-Clusters=4,5; Pyramid=3,5,6; Square=4)

• Comprehensive Learning PSO

With the aim of successfully tackling complex multimodal functions, Liang et al.(2006) [LQSB06] designed the Comprehensive Learning PSO (CLPSO), an interesting learning strategy adapted to PSO whereby all other particles' personal best information is used to update a particle's velocity. This strategy differs from FIPS in such a way that in CPSO different component dimensions (d) are informed by different personal best positions of neighbors, whereas in FIPS, each particle dimension is informed by all personal best position of involved particles in the neighborhood. In this way, the velocity updating equation used by CLPSO is as follows:

$$v_i^{t+1}(d) \leftarrow \omega \cdot v_i^t(d) + U^t[0, \varphi] \cdot (p^t_{f_i(d)}(d) - x_i^t(d)) \tag{3.12}$$

where $f_i = [f_i(1), f_i(2), \cdots, f_i(D)]$ defines which particles' personal best p the particle i should follow. Then, $p^t_{f_i(d)}(d)$ is the corresponding dimension of any particle's p including it own personal best, and the decision depends on probability Pc, referred to as the learning probability, which can

take different values for different particles. For each dimension d of particle i, a random number is generated. If this random number is higher than Pc_i, the corresponding dimension will learn from its own p_i; otherwise it will learn from another particle's personal best p. When this last happens, a tournament selection procedure is used to select the "neighbor" particle from which the current particle i learns.

Therefore, new positions of particles are generated using the information derived from different particle's personal best positions. To ensure that a given particle learns from good neighbors and to minimize the time wasted on poor directions, particles are allowed to learn from neighbors until the particle do not reach any improvement for a certain number of iterations called the refreshing gap m, then j_i is reassigned for that particle. CLPSO has been proven to be a prominent solver on complex multimodal and non-separable problem functions [LQSB06].

• Orthogonal Learning PSO

With the motivation of making an efficient use of search information from $\mathbf{p}$ to $\mathbf{b}$, Zhan et al.(2011) [ZZLS11] proposed the Orthogonal Learning PSO (OLPSO). This algorithm uses an Orthogonal Experimental Design (OED) method that offers an ability to discover the best combination levels for different factors with a reasonably small number of experimental samples. Then, OLPSO combines information of personal and social informants $\mathbf{p}$ and $\mathbf{b}$, respectively, to form an orthogonal guidance vector $\mathbf{o}$. The particle's velocity update rule is then as specified in Equation 3.13.

$$\mathbf{v}_i^{t+1} \leftarrow \omega \cdot \mathbf{v}_i^t + U^t[0, \varphi] \cdot (\mathbf{o}_i^t - \mathbf{x}_i^t) \tag{3.13}$$

The guidance vector $\mathbf{o}_i$ is constructed for each particle i from $\mathbf{p}_i$ and $\mathbf{b}_i$ as follows,

$$\mathbf{o}_i^t \leftarrow \mathbf{p}_i^t \oplus \mathbf{b}_i^t \tag{3.14}$$

where the symbol $\oplus$ stands for the OED operation. Therefore, with the resulting orthogonal learning vector $\mathbf{o}_i^t$, particle i adjusts its velocity and position every generation. This vector $\mathbf{o}_i^t$ is then kept constant until any improvement has not been reached. In the calculation of $\mathbf{o}_i^t$, each of the D dimensions is regarded as a factor and therefore, the are D factors in the OED. This results in $M = 2^{\lg_2(D+1)}$ orthogonal combinations, since the level of each factor is two. In this way, M is no longer than $2 \cdot D$, which is significantly smaller than 2^D, the total number of combinations.

The orthogonal learning strategy can be applied to any kind of topology structure, that is $\mathbf{b}$ can be taken as the global best, as well as the best particle in a local neighborhood. When $\mathbf{p}_i$ is the same as $\mathbf{b}_i$, OED makes no contribution, and OLPSO will randomly select another particle r to construct the new $\mathbf{o}_i^t$ vector. OLPSO have been applied to numerous benchmarking functions, as well as several real world problems showing a successful performance in practically all cases.

• Incremental Social Learning PSO

Incremental Social Learning (ISL) is a framework by means of which one agent is added to a given population at a time according to a schedule to reinforce the global knowledge of the population. Then, every time a new agent is added to the population, it should learn socially from a subset of the more experienced agents. Using this mechanism Montes de Oca et al.(2011) [MdOSVdED11] proposed the Incremental Social Learning PSO (IPSO). In this algorithm, every time a new particle is added, it is initialized using information from particles that are already part of the population. This mechanism is implemented as an initialization rule that moves the new particle $\mathbf{x}_{new}$ from

an initial randomly generated position in the problem's search space to one that is closer to the position of a particle that serves as a "model" $\mathbf{p}_{model}$ to imitate (usually the best particle in the swarm). The initialization rule used in IPSO is the following:

$$\mathbf{x}'_{new} \leftarrow \mathbf{x}_{new} + U \cdot (\mathbf{p}_{model} - \mathbf{x}_{new}) \tag{3.15}$$

where $U \in [0, 1)$ is a uniformly distributed random number, which is the same for all dimensions in order to ensure that the new particle's updated previous best position will lie somewhere along the direct attraction vector $\mathbf{p}_{model} - \mathbf{x}_{new}$. Using independent random numbers for each dimension would reduce the strength of the bias induced by the initialization rule because the resulting attraction vector would be rotated and scaled with respect to the direct attraction vector. Once the rule is applied, the new particle's previous best position is initialized to the point $\mathbf{x}'_{new}$ and its velocity is set to zero. Finally, the new particle's neighborhood, that is, the set of particles from which it will receive information in subsequent iterations, is generated at random, respecting the connectivity degree of the swarm's topology.

• Self Learning PSO

A recent version consist of the Self Learning PSO (SLPSO) proposed by Li et al.(2012) [LYN12], which is inspired by the idea of division labor. In SLPSO, each particle informed four different learning sources: the global best particle b, its personal best position p, the best position of a randomly chosen particle pr, and a random position r nearby. The four strategies play roles of convergence, exploitation, exploration, and jumping out of the basis of attraction of local optima, respectively. The corresponding learning equations to these four different situations are as follows:

- Operator **a**: learning from its personal best position
  exploitation:
$$\mathbf{v}_i^{t+1} \leftarrow \omega \cdot \mathbf{v}_i^t + U^t[0, \varphi] \cdot (\mathbf{p}_i^t - \mathbf{x}_i^t) \tag{3.16}$$

- Operator **b**: learning from the personal best position of a random neighbor
  exploitation:
$$\mathbf{v}_i^{t+1} \leftarrow \omega \cdot \mathbf{v}_i^t + U^t[0, \varphi] \cdot (\mathbf{p}r_i^t - \mathbf{x}_i^t) \tag{3.17}$$

- Operator **c**: learning from the global best position
  exploitation:
$$\mathbf{v}_i^{t+1} \leftarrow \omega \cdot \mathbf{v}_i^t + U^t[0, \varphi] \cdot (\mathbf{b}_i^t - \mathbf{x}_i^t) \tag{3.18}$$

- Operator **d**: learning from a random position nearby
  jumping out:
$$\mathbf{x}_i^{t+1} \leftarrow \omega \cdot \mathbf{x}_i^t + \mathbf{v}_{avg}^t \cdot N(0, 1) \tag{3.19}$$

In jumping step, $\mathbf{v}_{avg}$ is the average speed of all particles for each dimension, which is calculated by: $\mathbf{v}_{avg} = \frac{\sum_{i=1}^{Ss} |\mathbf{v}_i|}{Ss}$, where Ss is the swarm size. The choice of which learning option is the most suitable would depend on the shape of the local fitness landscape where a particle is located, which is achieved by means of an adaptive learning mechanism and updating a given selection ratio.

3.1.4 Related Approaches: Differential Evolution (DE)

Differential Evolution (DE) is an efficient metaheuristic for real parameter optimization problems that has been successfully used in a multitude of studies [PSL05] from its design by Storn and Price in 1995 [SP95]. DE is often compared against PSO in the literature, so it constitutes one of its direct competitors. Similarly to PSO, DE is easy to use and implement, and it also needs a few parameters to be tuned. Nevertheless, the main feature lies in the population of real-valued vectors that, as to in PSO, are combined by a series of differential equations to update them on the search landscape.

Therefore, in DE, the task of generating new individuals (real-valued vectors) is performed by operators such as the differential mutation (also known as "perturbation") and crossover. A *mutant individual* $\mathbf{w}_{g+1}^{i}$ can be generated by a perturbation scheme selected to initially construct the algorithm. Storn and Price [SP95] proposed four original perturbation schemes:

- *DE/rand/1*:
$$\mathbf{w}_i^{t+1} \leftarrow \mathbf{x}_{r1}^t + F \cdot (\mathbf{x}_{r2}^t - \mathbf{x}_{r3}^t) \tag{3.20}$$
- *DE/best/1*:
$$\mathbf{w}_i^{t+1} \leftarrow \mathbf{b}^t + F \cdot (\mathbf{x}_{r1}^t - \mathbf{x}_{r2}^t) \tag{3.21}$$
- *DE/best/2*:
$$\mathbf{w}_i^{t+1} \leftarrow \mathbf{b}^t + F \cdot (\mathbf{x}_{r1}^t + \mathbf{x}_{r2}^t - \mathbf{x}_{r3}^t - \mathbf{x}_{r4}^t) \tag{3.22}$$
- *DE/rand-to-best/1*:
$$\mathbf{w}_i^{t+1} \leftarrow \mathbf{x}_i^t + \lambda \cdot (\mathbf{b}^t - \mathbf{x}_i^t) + F \cdot (\mathbf{x}_{r1}^t - \mathbf{x}_{r2}^t) \tag{3.23}$$

where $r1, r2, r3, r4 \in \{1, 2, \ldots, i-1, i+1, \ldots, N\}$ are random integers mutually different, and also different from the index i. The mutation constant $F > 0$ stands for the amplification of the difference between the individuals $\mathbf{x}_{r's}$ and it avoids the stagnation of the search process. The last parameter, λ controls the greediness of the fourth scheme. To reduce the number of control variables, these two last parameters are usually set to the same value $F = \lambda$.

In order to increase even more the diversity in the population, each mutated individual undergoes a crossover operation with the *target individual* $\mathbf{x}_i^t$, by means of which a *trial individual* $\mathbf{u}_i^{t+1}$ is generated. A randomly chosen position is taken from the mutant individual to prevent that the trial individual replicating the target individual.

$$u_i^{t+1}(j) \leftarrow \begin{cases} w_i^{t+1}(j) & \text{if } r(j) \leq Cr \text{ or } j = j_r, \\ x_i^t(j) & \text{otherwise.} \end{cases} \tag{3.24}$$

As shown in Equation 3.24, the crossover operator randomly chooses a uniformly distributed integer value j_r and a random real number $r \in [0, 1]$, also uniformly distributed for each component j of the trial individual $\mathbf{u}_i^{t+1}$. Then, the crossover probability Cr, and r are compared just like j and j_r. If r is lower or equal than Cr (or j is equal to j_r) then we select the j^{th} element of the mutant individual to be allocated in the j^{th} element of the trial individual $\mathbf{u}_i^{t+1}$. Otherwise, the j^{th} element of the target individual $\mathbf{x}_i^t$ becomes the j^{th} element of the trial individual.

Finally, a selection operator decides on the acceptance of the trial individual for the next generation if and only if it yields a reduction (assuming minimization) in the value of the fitness function $f()$, as shown by the following Equation (3.25):

$$\mathbf{x}_i^{t+1} \leftarrow \begin{cases} \mathbf{u}_i^{t+1} & \text{if } f(\mathbf{u}^{t+1}i) \leq f(\mathbf{x}_i^t), \\ \mathbf{x}_i^t & \text{otherwise.} \end{cases} \tag{3.25}$$

Algorithm 3 shows the pseudocode of DE. After initializing the population, the individuals evolve during a number of iterations ($MAXIMUM_t$). Each individual is then mutated (Line 5) and recombined (Line 6). The new individual is selected (or not) following the operation of Equation 3.25 (Lines 7 and 8).

Algorithm 3 Pseudocode of Canonical DE

```
1: P ←initializePopulation(Ps)
2: while t < MAXIMUM_t do
3:    for each individual i of P do
4:       choose mutually different r_s values
5:       w_i^{t+1} ← mutation(x_{r_s}^t, F)   //Eq. 3.20 or Eq. 3.21 or Eq. 3.22 or Eq. 3.23
6:       u_i^{t+1} ← crossover(x_i^t, w_i^{t+1}, Cr)   //Eq. 3.24
7:       evaluate(u_i^{t+1})
8:       x_i^{t+1} ← selection(x_i^t, u_i^{t+1})   //Eq. 3.25
9:    end for
10: end while
```

Although there is no precise indication of which scheme is the best (it seems to depend to the tackled problem), nor there is a definitive indication of which values to use for parameters, Storn and Price [PSL05] proposed a series of empirically tested values: crossover probability $Cr \in [0, 1]$ must be considerably lower than 1, e. g., 0.3. Nevertheless, if no convergence can be achieved, a value in $[0, 0.8]$ should be used; for many applications, a population size of $P = 10 \cdot D$ is a good choice, being D is the problem dimension; value F is usually chosen in $[0.5, 1]$, in such a way that, the higher the population size, the lower the weighting factor F.

3.2 A General Survey on PSO Applications

This section presents a pragmatic review of the impact of PSO publications in the literature. Our aim here is to organize and extract underlying information concerning the challenges and opportunities offered by this technique. Then, following the general structure of this thesis, on the first place, we made an analysis of research studies on which standard benchmarks of continuous optimization functions are used to assess PSO approaches. We focus on the main properties of most relevant benchmarks and their adaptation to the specific features of the particle swarm algorithm. On the second place, we made an extensive catalogue of applications tackled with PSO in the literature. We count over more than ten thousands publications on this metaheuristic stored at digital libraries and we summarize them according to different categories of application domains.

3.2.1 Benchmarking

In the last two decades, the performance of PSO has mainly been assessed on subsets of popular benchmark of continuous optimization problems such as the Sphere, Schaffer's, Griewank's, Ackley's, Rosenbrock's, and Rastrigin's functions showing promising results. These functions can be characterized by the number of optima (modality), the degree of convexity, the existence of plateaus, the ruggedness, and the dependence of variables. Nevertheless, other features like the number of funnels, the rotation, and the shifting to different bias to the global optimum are not considered in such sets of functions. In addition, a great number of comparisons made in past research studies are often confusing and limited to the test problems used by them. Furthermore,

these function problems are in general well adapted to the operation of PSO, and do not stress the intrinsic deficiencies of this algorithm properly.

A series of benchmark test suites have appeared that provide well defined procedures to evaluate and compare algorithms on continuous optimization. Two first attempts can be found in De Jongs' functions [DJ75] (DEJONG'75) and Yao's set [YLL99] (XYAO'99). These two test suites have largely been used on evolutionary and stochastic local search techniques, although they present two major concerns: the global optimum is generally located at the origin of coordinates, and the distribution of local optima is usually regular with separable variables. All these features make the sets of functions unsuitable to test PSO approaches. More recently, new benchmarks are appearing that solve these and other issues commented before. They are becoming popular from the Special Session of Continuous Optimization of CEC'2005 [SHL$^+$05] (CEC'05), from which, several proposals have been worked from different points of view: large scale capabilities [TYS$^+$07] [TLS$^+$10] (CEC'08 and CEC'10), noisy-noiseless functions [HAFR09a] (BBOB'09), scalability studies [HLM10b] (SOCO'10), and real-world problems [DS11](CEC'11). In addition, subsets of CEC'05 functions have been directly used to constitute the test suites in other special sessions like MAEB'09 [GNAAL09] (used in Chapter 4), where only multimodal functions were experimented (MAEB'09 $\subset$ CEC'05). These test suites contain functions with different properties: single and multi-funnel, shifted global optimum, rotation, dependency of variables, etc.; which make them highly appropriate to find deficiencies in PSO, and to compare it against other techniques in the state of the art.

In this sense, another important criterion when selecting a benchmark test suite consists on its popularity in the current literature. In this way, we could know before-hand the number of techniques tested with a given benchmark set and what are the best results to beat that constitute the current state of the art. With this aim, we have carried out a revision of around 400 PSO research studies. We limited our study to the following well-known conferences: GECCO, CEC, PPSN, and SIS[1]; and journals: TEC, SMC, EC, SI, SOCO, and IS[2]; in the area. From these, we filtered 202 contributions concerning versions of PSO tested with benchmark functions on continuous optimization.

As a result, Figure 3.3 plots a bar graph representing the percentage of contributions in which a PSO approach is, at least, evaluated with an academic benchmark. In this figure, we can also see the proportion of these contributions with regards to the number of years each test suite has been kept in use. Element OTHERS in bar graph includes all contributions concerning PSO using "miscellaneous" sets of functions (from the introduction of PSO in 1995 to nowadays, 2012). At the moment, CEC'11 test suite has not been used to evaluate any PSO version.

We can easily observe in Figure 3.3 that CEC'05 is the most used test suite for assessing PSO approaches. In spite of coexisting with other modern benchmarks, the use of CEC'05 is increasing along with the years. In addition, there exist several works devoted to characterize the fitness landscape of CEC'05 functions by means of the fitness distance correlation [MS11], and the mean dispersion [MBS09], so then we can classify the degree of difficulty they might present. In this sense, CEC'05 satisfies almost all desirable features to evaluate PSO approaches. However, a disadvantage can be found in this benchmark concerning the restriction of working with a relative short dimensionality, as only a maximum of 50 variables can be used. Therefore, features derived from the scalability can not be properly studied with CEC'05 (similar happens with BBOB'09).

[1]Genetic and Evolutionary Computation Conference (GECCO), Congress on Evolutionary Computation (CEC), Parallel Problem Solving From Nature (PPSN), and Swarm Intelligence Symposium (SIS)

[2]IEEE Transactions on Evolutionary Computation (TEC), IEEE Transactions on System Man and Cybernetics (SMC), Evolutionary Computation Journal (EC), Swarm Intelligence (SI), Soft Computing (SOCO), and Information Sciences (IS)

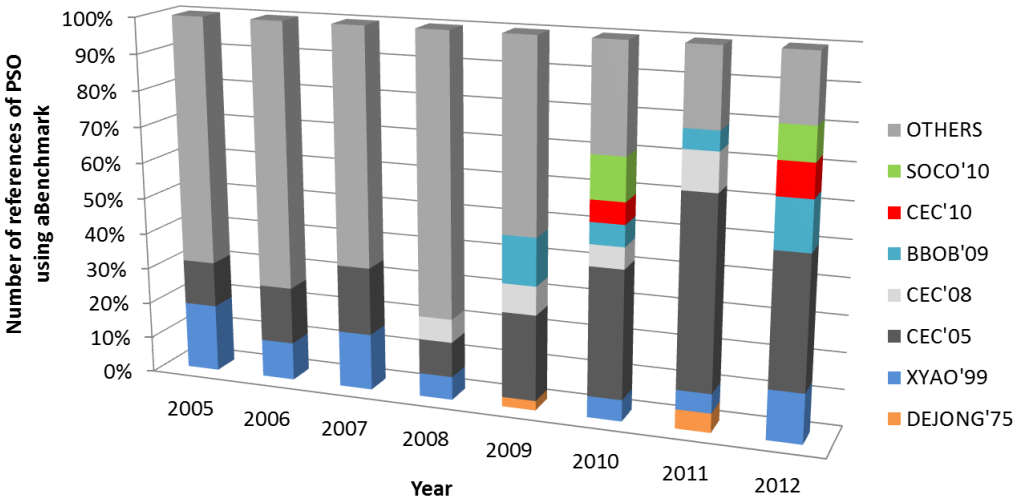


Figure 3.3: Percentage of contributions in which any PSO approach is evaluated with an academic benchmark

Other benchmarks such as CEC'08, SOCO'10, CEC'10 and CEC'12, offer the chance of working with large scale problem dimensions, since just up to one thousand variables can be tackled. They are all extended versions of CEC'08 (CEC'08 $\subset$ SOCO'10, CEC'08 $\subset$ CEC'10 and CEC'10 = CEC'12) and they have been widely used in the literature, also offering available results [3] to be compared with. In this way, a recommendable practice in this field might be using CEC'05 plus another set of large scale functions (e. g., CEC'08 or SOCO'10), hence covering the complete spectrum of hard properties to asses PSO approaches, as well as other metaheuristics for continuous optimization.

Table 3.2 shows the set of functions mostly used in this thesis work (Chapters 5 and 7) with their most interesting features: unimodal, multimodal, separable, non-separable, shifted to biased optimum, rotated, and hybrid composed. The respective bounds of search ranges and biases to optima are also indicated. The detailed descriptions of all these functions can be found in [HLM10b], and [SHL$^+$05]. The first part of this table correspond to 19 functions (labeled *soco*$*$) provided in SOCO'10. From this benchmark, functions *soco*1 to *soco*6 were originally used in CEC'08 [TYS$^+$07]. Functions *soco*7 to *soco*11 were added to the first ones in the special session of ISDA'09 [HL09a], and functions *soco*12 to *soco*19 consist on hybridized functions that combine two others (being one of them non-separable, at least). The second part of Table 3.2 includes 21 more functions proposed in CEC'05 (labeled as *cec*$*$) to the previous 19 of SOCO'10, then constituting a set of 40 functions. We have to notice that, as done in [LMdOA$^+$11], from the original 25 functions of CEC'05 we omitted *cec*1, *cec*2, *cec*6, and *cec*9, since they are the same as *soco*1, *soco*3, *soco*4, and *soco*8.

[3] [online] `http://sci2s.ugr.es/EAMHCO/`

Table 3.2: SOCO'10 and CEC'05 benchmark test suites with functions' features: unimodal, multimodal, separable and non-separable, rotated and non-rotated

f	Name	Uni./Mul.	Sep.	Rot.	Search Range	f^*
soco1	Shifted Sphere	U	Y	N	[-100, 100]	-450
soco2	Shifted Schwefel 2.21	U	S	N	[-100, 100]	-450
soco3	Shifted Rosenbrock	M	N	N	[-100, 100]	390
soco4	Shifted Rastrigin	M	Y	N	[-5, 5]	-330
soco5	Shifted Griewank	M	N	N	[-600, 600]	-180
soco6	Shifted Ackley	M	Y	N	[-32, 32]	-140
soco7	Shifted Schwefel 2.22	U	Y	N	[-10, 10]	0
soco8	Shifted Schwefel 1.2	U	N	N	[-65.536, 65.536]	0
soco9	Shifted Extended f10	U	N	N	[-100, 100]	0
soco10	Shifted Bohachevsky	U	N	N	[-15, 15]	0
soco11	Shifted Schaffer	U	N	N	[-100, 100]	0
soco12	Hybr. Comp. soco9 $\bigoplus_{0.25}$ soco1	M	N	N	[-100, 100]	0
soco13	Hybr. Comp. soco9 $\bigoplus_{0.25}$ soco3	M	N	N	[-100, 100]	0
soco14	Hybr. Comp. soco9 $\bigoplus_{0.25}$ soco4	M	N	N	[-5, 5]	0
soco15	Hybr. Comp. soco10 $\bigoplus_{0.25}$ soco7	M	N	N	[-10, 10]	0
soco16	Hybr. Comp. soco9 $\bigoplus_{0.50}$ soco1	M	N	N	[-100, 100]	0
soco17	Hybr. Comp. soco9 $\bigoplus_{0.75}$ soco3	M	N	N	[-100, 100]	0
soco18	Hybr. Comp. soco9 $\bigoplus_{0.55}$ soco4	M	N	N	[-5, 5]	0
soco19	Hybr. Comp. soco10 $\bigoplus_{0.75}$ soco7	M	N	N	[-10, 10]	0
cec3	Shifted Rotated High Conditioned Elliptic	U	S	R	[-100, 100]	-450
cec4	Shifted Schwefel's Problem 1.2 with Noise	U	S	N	[-100, 100]	-450
cec5	Schwefel's Problem 2.6	U	S	N	[-100, 100]	-310
cec7	Shif. Rot. Griewank's. G.O. Outside of Bounds	M	S	R	[0, 600]	-180
cec8	Shif. Rot. Ackley's with G.O. on Bounds	M	S	R	[-32, 32]	-140
cec10	Shifted Rotated Rastrigin's	M	S	R	[-5, 5]	-330
cec11	Shifted Rotated Weierstrass	M	N	R	[-0.5, 0.5]	90
cec12	Schwefel's Problem 2.13	M	N	N	[$-\pi$, π]	-460
cec13	Shifted Expanded Griewank's plus Rosenbrock's	M	N	N	[-3, 1]	-130
cec14	Shifted Rotated Expanded Scaffer's F6	M	S	R	[-100, 100]	-300
cec15	Hybrid Composition (f1-f2,f3-f4,f5-f6,f7-f8,f9-f10)	M	N	N	[-5, 5]	120
cec16	Rotated Version of Hybrid Composition f15	M	N	R	[-5, 5]	120
cec17	F16 with Noise in Fitness	M	N	R	[-5, 5]	120
cec18	Rot. Hybr. Comp. (f1-f2,f3-f4,f5-f6,f7-f8,f9-f10)	M	N	R	[-5, 5]	10
cec19	Rot. Hybr. Comp. Narrow Basin Global Optimum	M	N	R	[-5, 5]	10
cec20	Rot. Hybr. Comp. G. O. on Bounds	M	N	R	[-5, 5]	10
cec21	Rot. Hybr. Comp. (f1-f2,f3-f4,f5-f6,f7-f8,f9-f10)	M	N	R	[-5, 5]	360
cec22	Rot. Hybr. Comp. High Condition Number Matrix	M	N	R	[-5, 5]	360
cec23	Non-Continuous Rotated Hybrid Composition	M	N	R	[-5, 5]	360
cec24	Rot. Hybr. Comp. (f1,f2,f3,f4,f5,f6,f7,f8,f9,f10)	M	N	R	[-5, 5]	260
cec25	Rot. Hybr. Comp. G. O. Outside of Bounds	M	N	R	[2, 5]	260

3.2.2 Real World Applications

Particle swarm optimization has been used across a wide range of real world applications. Areas where PSO approaches have shown to be powerful solvers include multimodal problems, black-box simulations, parameter tuning applications, and problems for which there are no specialized methods available or all specialized methods show unsatisfactory results.

PSO applications are so numerous and diverse and just to review a subset of the most paradigmatic ones involves a huge deal of work. An affordable way to review the state of the art consists on listing the main application areas where PSOs have been successfully used. Therefore, based on previous PSO overviews [PKB07, SM09] and after a thorough search on important digital libraries on the area: IEEExplore and ACM Digital Library, we have divided applications into 28 different categories, although many applications span more than one category. We have based our categorization on an analysis of over PSO publications for the last six years, that is, we started on 2006 to continue where those previous overviews finished [PKB07, SM09]. Because of the large number of applications reviewed, here we will provide our categorization without citations.

Table 3.3: PSO publications by year

Application domain	2006	2007	2008	2009	2010	2011	2012	T_{domain}
Image and video analysis	1	3	3	40	10	13	2	72
Electricity networks and load dispatching	0	0	3	4	3	2	3	15
Control systems	33	27	54	64	65	72	68	383
Electronics and electromagnetics	9	6	8	10	33	87	45	198
Antenna design	47	49	46	72	88	70	88	460
Scheduling	40	49	91	121	109	81	84	575
Industrial design	35	33	69	88	67	58	37	387
Communication networks	204	248	377	795	1263	1204	1107	5198
Chemical processes	0	0	0	0	2	1	2	5
Biomedical	1	6	14	18	17	16	5	77
Clustering, classification and data mining	39	50	90	110	117	88	93	587
Fuzzy and neuro systems and control	3	2	2	1	4	3	3	18
Signal processing	161	194	269	358	663	875	793	3313
Neural networks	94	121	183	254	207	122	60	1041
Robotics	256	216	366	358	521	606	581	2904
Prediction and forecasting	3	3	13	22	35	16	14	106
Diagnosis, detection, and recovering of faults	14	4	12	40	44	11	14	139
Sensors and sensor networks	26	37	57	77	76	81	98	452
Computer graphics and visualization	0	2	1	4	2	1	0	10
Engines and electrical motors	2	0	1	3	1	0	1	8
Metalurgy materials	2	0	1	2	3	1	0	9
Music	1	0	0	2	5	4	1	13
Games	6	5	6	10	10	10	6	53
Security and military	1	1	0	1	0	0	0	3
Finance and economics	1	1	0	7	2	0	2	13
Transportation	42	54	69	119	159	365	291	1099
Traffic management	2	3	5	14	8	3	3	38
Vehicular networks	3	2	3	5	3	11	1	28
T_{year}	1026	1116	1743	2599	3517	3801	3470	17204

The main PSO application categories are presented in Table 3.3, where the number of publications on different domains are organized by year, from 2006 to 2012. As a summary of this table, in Figure 3.4 the global count of publications is graphically shown by domain category and year in logarithmic scale. Application domains like: Communication Networks (with 5198 r.p.p. [4]), Signal Processing (3313 r.p.p.), Robotics (2904 r.p.p.), Neural Networks (1041 r.p.p.) and Transportation (1099 r.p.p.) concentrate most of research efforts in the last years, whereas other domains like: Chemical Processes (3 r.p.p.), Security and Military (3 r.p.p.), Engines and Electrical Motors (8 r.p.p.), Metallurgy and Materials (9 r.p.p.), and Computer Graphics and Visualization (10 r.p.p.) have been shortly worked by industrial and research community in the context of particle swarm optimization.

In general, the number of PSO applications is growing over the years (2012 is still in study). Nevertheless, Table 3.3 shows how for certain domains, the use of PSO is becoming popular while other research areas are reducing the use of this algorithm. In this sense, for some categories: Transportation, Robotics, Signal Processing, Electronics and Electromagnetics, Sensor Networks, Vehicular Networks, and Control Systems, the number of research publications increases, specially from 2010 to 2012. However, we can also observe that traditional applications of PSO like: Neural Networks, Clustering and Classification in Data Mining, Image Analysis and Scheduling are being less worked from the last three years.

As a summary, we can state that this algorithm is the center of a big deal of developments and applications, clearly worth of further analysis and plenty of opportunities for competitive research in the upcoming years.

[4]Research paper publications

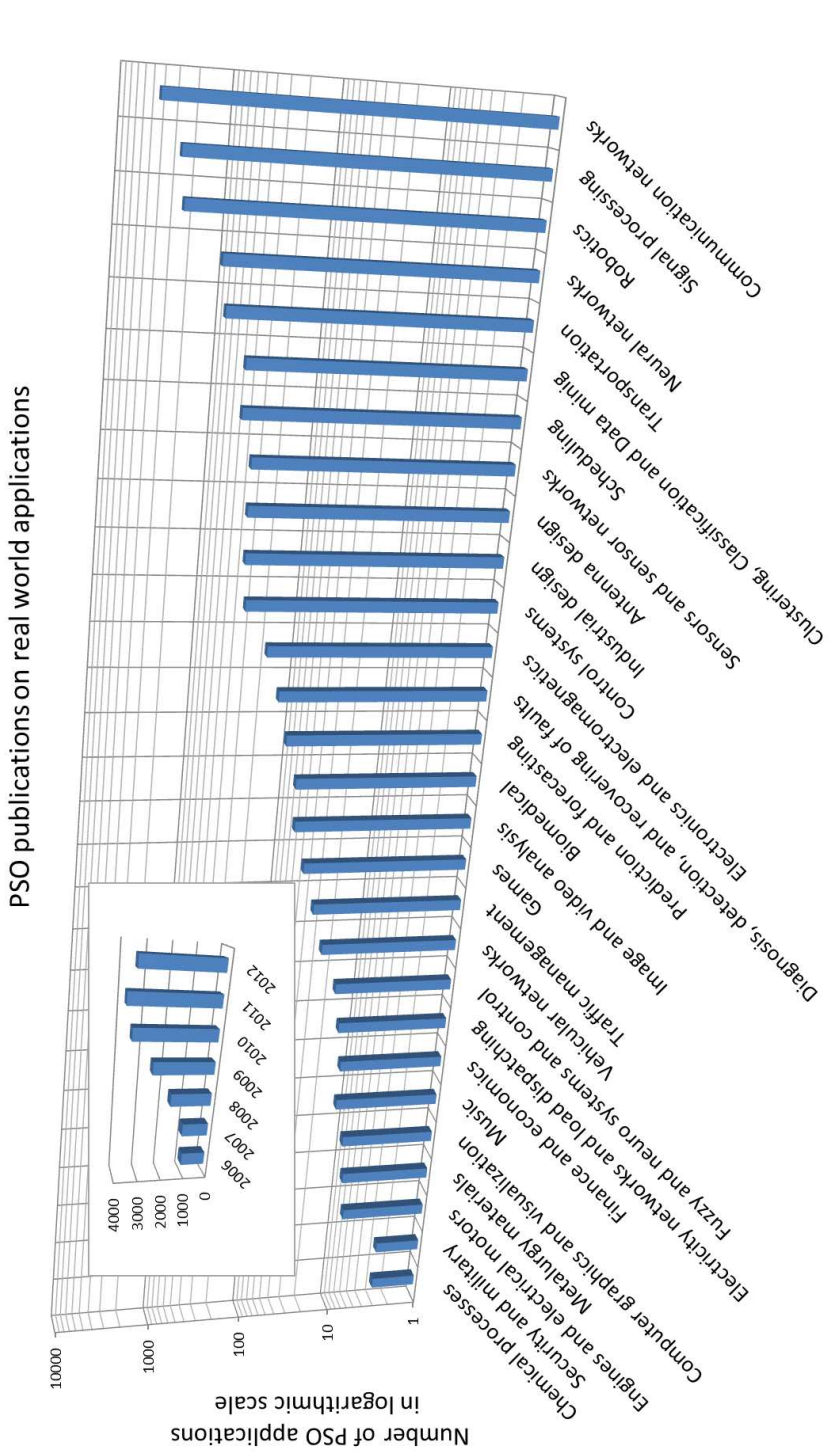


Figure 3.4: Summary of PSO publications on real world applications by domain. A subgraph of these publications by year is also plotted

Part II

Algorithm Proposals and Validation on Benchmarks

Chapter 4

DEPSO: Hybrid PSO with DE

4.1 Introduction

As we commented in Chapter 3, Particle Swarm Optimization and Differential Evolution have been widely used on numerous and heterogeneous optimization problems, being these two algorithms specially well adapted to real solution encoding. For this reason, several approaches have appeared in the past [XJZ+12] that combine them, since it represents a promising way to create more powerful optimizers. A first attempt was by Das *et al.* [DAK08], where an initial hybridization of PSO and DE reached successful results in comparison with both, PSO and DE in their canonical formulations, on a set of benchmark functions. Recently, a comprehensive overview of PSO-DE hybridizations have been presented in [XJZ+12], where more than 45 proposals are studied, with their applications to different research areas like: clustering and classification, economic dispatch, image processing, constraints handling, robotics, and reactive power.

In this chapter, we are aimed at performing a through experimentation of our initial proposal of PSO hybridized with DE, called DEPSO [GNAAL09, GNAA09a, GNAA09b], to discover its intrinsic optimizing potentials in comparison with other advanced optimizers. Our DEPSO uses the differential variation scheme employed in DE for adjusting the velocity of PSO's particles. In this way, by combining search strategies and differential operators present in PSO and DE, we expected to improve the performance of the resulting technique. Our motivation is to try to create a single hybrid that can be claimed to be the state of the art to guide future research in contrast to the many past approaches validated very partially in the literature.

For this task, we have focused on three different procedures presented in two scientific conferences in the area of metaheuristics. On the first place, we have followed the test suite proposed in the Special Session of Continuous Optimization of MAEB'09 [HL09b], which is in turn based on the function test set proposed in CEC'05 [SHL+05]. On the second place, we have also followed the experimental framework proposed in the Special Session of Real Parameter Optimization of GECCO'09 [HAFR09a], according to the Black-Box Optimization Benchmarking (BBOB) for noisy and noiseless functions [HFRA09a, HFRA09b]. We have performed the two complete procedures with different problem dimensions (D): from 2 to 40 variables. Our proposal is shown to obtain an accurate level of coverage rate in comparison with other relevant techniques, even for the higher dimensions, and despite the relatively small number of function evaluations used ($1000 \cdot D$).

The remaining of the chapter is organized as follows. In Section 4.2 our DEPSO is described. Sections 4.3 and 4.4 present the experiments and results obtained with regards to MAEB'09 and

BBOB'09 procedures, respectively. In Section 4.5, an analysis of CPU Timing consumption is reported. Finally, conclusions and remarks are given in Section 4.6.

4.2 The Algorithm: DEPSO

Our proposal basically consists in running a PSO algorithm in which we incorporate ideas of DE. For each particle $\mathbf{p}$ (with velocity and position vectors $\mathbf{v}$ and $\mathbf{x}$, respectively) the velocity is updated according to two main influences: social and differential variation factors. The social factor concerns the local or global best position $\mathbf{g}$ of a given neighborhood of particles (being global when this neighborhood is the entire swarm). The differential variation factor is composed by using two randomly selected particle positions as made in DE. This way, for each particle $\mathbf{x}_i$ of the swarm, a differential vector $(\mathbf{x}^t_{r1} - \mathbf{x}^t_{r2})$ is generated following the *DE/rand/1* scheme of canonical DE, by mean of which, particles $\mathbf{x}^t_{r1}$ and $\mathbf{x}^t_{r2}$ are randomly (uniformly) selected (with $i^t \neq r1^t \neq r2^t$) from the whole swarm. Then, the new velocity $\mathbf{v}^{t+1}_i$ of a particle i is calculated according to the following equation:

$$\mathbf{v}^{t+1}_i \leftarrow \omega \cdot \mathbf{v}^t_i + F \cdot (\mathbf{x}^t_{r1} - \mathbf{x}^t_{r2}) + U^t[0, \varphi_i] \cdot (\mathbf{b}^t_i - \mathbf{x}^t_i) \tag{4.1}$$

where ω is the inertia weight of the particle that controls the trade-off between global and local experience. Perturbation constant $F = U(0, 1)$ is a scaling factor applied to the differential vector and stands for the amplification of the difference between the individuals: it avoids the stagnation of the search process. The third component corresponds to the social factor which is influenced by the global best $\mathbf{g}$ of the swarm and directly proportional to the social coefficient φ. Therefore, in the velocity calculation, the standard personal influence used in PSO is replaced in our proposal by the differential vector $F \cdot (\mathbf{x}^t_{r1} - \mathbf{x}^t_{r2})$.

Similarly to DE, the update of each j component of the velocity vector of a given particle i is carried out by means of a crossover operation with the target velocity v^{t+1}_i to generate the trial particle u^{t+1}_i as specified in Equation 4.2.

$$u^{t+1}_i(j) = \begin{cases} v^{t+1}_i(j) & \text{if } r^t(j) \leq Cr \text{ or } j = j_r, \\ v^t_i(j) & \text{otherwise.} \end{cases} \tag{4.2}$$

Here, $r \in [0, 1]$ is a uniformly distributed value which determines if the j^{th} component is selected from the new velocity or is selected from the current velocity, based on the crossover probability $Cr \in [0, 1]$. This mechanism is used to increase the exploitation ability of the algorithm, mostly when tackling with non-separable fitness landscapes [PSL05]. Finally, a particle i updates its position through the trial particle $(\mathbf{u}^{t+1}_i)$, only if the new one $\mathbf{x'}^t_i$ is lower or equal than the current position $\mathbf{x}^t_i$ (assuming minimization). In other case, the particle keeps its current position (see equations 4.4 and 4.3).

$$\mathbf{x}^{t+1}_i = \begin{cases} \mathbf{x'}^t_i & \text{if } f(\mathbf{x'}^t_i) \leq f(\mathbf{x}^t_i) \\ \mathbf{x}^t_i & \text{otherwise,} \end{cases} \tag{4.3}$$

being

$$\mathbf{x'}^t_i \leftarrow \mathbf{x}^t_i + \mathbf{u}^{t+1}_i \tag{4.4}$$

Additionally, with certain probability p_{mut}, a mutation operation is made on each particle in order to avoid an early convergence to a local optimum. The new mutated position $\mathbf{x}^{t+1}$ is generated by means of the Equation 4.5.

$$\mathbf{x}^{t+1} \leftarrow \mathbf{x}_{low} + U(0,1) \cdot (\mathbf{x}_{upp} - \mathbf{x}_{low}) \tag{4.5}$$

Vectors $\mathbf{x}_{low}$ and $\mathbf{x}_{upp}$ correspond to lower and upper limits of each dimension of the function to optimize, respectively.

Algorithm 4 Pseudocode of DEPSO

```
1: S ← initializeSwarm(Ss)
2: while t < MAXIMUM_t do
3:    for each particle i of S do
4:       choose mutually different r1 and r2 values
5:       v_i^{t+1} ← velocityUpdate(ω, v_i^t, F, x_{r1}^t, x_{r2}^t, φ, b_i^t, x_i^t)   //Eq. 4.1
6:       u_i^{t+1} ← crossover(v_i^{t+1}, v_i^t, Cr, j_r)   //Eq. 4.2
7:       x'^t_i ← positionUpdate(x'^t_i, u_i^{t+1})   //Eq. 4.4
8:       evaluate(x'^t_i)
9:       x_i^{t+1} ← selection(x'^t_i, x_i^t)   //Eq. 4.3
10:      x_i^{t+1} ← mutation(p_mut, x_upp, x_low)   //Eq. 4.5
11:   end for
12: end while
```

Algorithm 4 shows the pseudocode of our hybrid DEPSO. First, particles in the swarm S are initialized as specified in [HAFR09a] (line 1). After this, each iteration step, each particle's velocity is updated (in line 5) applying differential vector perturbation and global best strategy. Then, crossover and particle's position update operations are performed in lines 6 and 7, respectively. After evaluating the new moved particle in line 8, the selection operation is carried out (line 9). Eventually, a mutation operation is performed depending on p_{mut} in line 10. Finally, the algorithm returns the best solution (particle's position) found during the whole process.

4.3 Experiments on MAEB'09

In this section, experiments and results concerning the evaluation of DEPSO on MAEB'09 are described and analyzed. We have implemented our proposal in C++ using the skeleton scheme and classes provided by the MALLBA library [ALGN+07] of metaheuristics. For the sake of a fair experimentation, we have to notice that, the same implementation of DEPSO, parameter setting and hardware execution platform have been used for MAEB'09 experiments, as well as for GECCO'09 ones (in the following section). The set of parameters used in DEPSO were selected after preliminary runs resulting successful fine-tuned combination as shown in Table 4.1. Independent runs were performed on a CONDOR [TTL05] middleware platform with Pentium IV 2.4 GHz processors, 1GB RAM memory and Linux O.S.

The set of functions used in this experimentation consists of 20 multimodal functions chosen from CEC'05 benchmark (defined in Chapter 3), from f_6 to f_{25}, with different properties of: rotation, shifted optima to the origin of coordinates, non-separable variables and composition with other complex functions. For this particular special session were considered two different problem dimensions: 10 and 30 variables. As specified in CEC'05 benchmark procedure, a number of 25 independent runs have been performed by DEPSO for each function and for each problem dimensionality. The stop condition has been established to a maximum number of $100,000 \cdot D$ function evaluations to perform, or when a fitness error lower than 1.0E-08 is reached.

Table 4.1: Parameter setting used in DEPSO

Description	Parameter	Value
Swarm size	Ss	20
Crossover probability	Cr	0.9
Inertia weight	ω	0.1
Differential scaling factor	F	$U(0,1)$
Social coefficient	φ	2.05
Mutation probability	p_{mut}	$\frac{1}{DIM}$

4.3.1 MAEB'09: Results and Analysis

Tables 4.2 and 4.3 contain the error values obtained by our DEPSO concerning functions: f_6 to f_{15}, and f_{16} to f_{25}, respectively for dimension $D = 10$. These results are shorted in tables from best to worst error values, being: 1^{st} (Best), 7^{th}, 13^{th} (Median), 19^{th}, and 25^{th} (Worst), and they are recorded at 1,000, 10,000, and 100,000 objective function evaluations. In addition, Means and Standard Deviations are also showed. Tables 4.4 and 4.5 contain the resulted error values for dimension $D = 30$. In this way, results are arranged as specified in CEC'05 (and recommended in MAEB'09) to facilitate the analysis and comparison against other algorithms, at different stages of the optimization process. Following this protocol, Figure 4.1 shows the different plots generated from execution traces of DEPSO's 13^{th} runs (Median) concerning all benchmark functions with dimension $D = 30$. Each curve shows the best fitness value at each iteration step of the 300,000 total evaluations.

For the analysis of results, we have compared the mean of error values (out of 25 independent runs) obtained by DEPSO against those mean errors of three reference algorithms of CEC'05 [SHL$^+$05], that were suggested at MAEB'09 for comparing our proposal. These algorithms are: Restarting and Increasing Population Covariance Matrix Adaptation Evolution Strategy G-CMA-ES [AH05], A Population-Based Steady-State Genetic Algorithm K-PCX [STD05] and Canonical Differential Evolution for Real Parameter Encoding DE [RKP05].

For this comparative analysis, we have used the statistical procedure of non-parametric tests as detailed in [GMLH09], since as shown in this paper, the set of functions considered for the experimentation does not fulfill at least one of the three conditions of: independency of samples, normality of distributions and/or heteroscedasticity, required for the application of parametric tests. In this way, we have considered the use of a Signed Ranking Wilcoxon test [Wil87], by mean of which, we compare our DEPSO against each one of previously cited algorithms. We have opted to use a confidence level of 95% in tests, meaning that if the resulted p-value is lower than 0.05, then the null hypothesis is rejected, and the algorithm with the higher value is the best one, with statistical significance.

Table 4.6 shows the results of applying the Wilcoxon test to the resulted error values of our proposal versus the ones of: G-CMA-ES, K-PCX and DE, for dimensions 10 and 30. When comparing a pair of techniques, for example, DEPSO and G-CMA-ES, value $R+$ indicates the average ranking where algorithm G-CMA-ES obtained mean error values ($f(\mathbf{x}) - f(\mathbf{x}^*)$) lower than the ones of DEPSO. Indicator $R-$ shows the average ranking where DEPSO obtains lower mean error values than G-CMA-ES. In this way, as we can observe in Table 4.6, for dimension $D = 10$, DEPSO obtains better mean ranking than DE and K-PCX. For dimension 30, DEPSO obtains better mean ranking than DE and G-CMA-ES. Nevertheless, the null hypothesis can not be rejected in any case, and therefore, we can not assure that distribution of results are different.

Table 4.2: Error values for functions f_6 to f_{15} with dimension $D = 10$ and 100,000 evaluations

Evals.	Exec. Num.	$\mathbf{f_6}$	$\mathbf{f_7}$	$\mathbf{f_8}$	$\mathbf{f_9}$	$\mathbf{f_{10}}$	$\mathbf{f_{11}}$	$\mathbf{f_{12}}$	$\mathbf{f_{13}}$	$\mathbf{f_{14}}$	$\mathbf{f_{15}}$
	1(Best)	4,71E+04	3,12E+00	2,05E+01	3,84E+01	3,53E+01	7,93E+00	2,66E+03	3,93E+00	3,73E+00	3,43E+02
	7	1,70E+05	7,11E+00	2,06E+01	4,59E+01	4,80E+01	1,05E+01	6,70E+03	5,31E+00	4,03E+00	4,81E+02
	13(Median)	2,33E+05	8,04E+00	2,06E+01	4,90E+01	5,58E+01	1,12E+01	9,50E+03	5,78E+00	4,14E+00	5,45E+02
1E+03	19	3,49E+05	1,11E+01	2,07E+01	5,27E+01	6,22E+01	1,19E+01	1,17E+04	6,21E+00	4,25E+00	5,99E+02
	25(Worst)	5,40E+05	1,21E+01	2,08E+01	5,95E+01	6,57E+01	1,22E+01	1,81E+04	6,50E+00	4,31E+00	6,16E+02
	Mean	2,68E+05	8,26E+00	2,07E+01	4,93E+01	5,48E+01	1,11E+01	9,39E+03	5,60E+00	4,11E+00	5,35E+02
	Std. Dev.	1,41E+05	2,60E+00	9,05E-02	5,92E+00	8,09E+00	9,67E-01	4,42E+03	6,81E-01	1,74E-01	7,50E+01
	1(Best)	5,40E-02	1,00E-02	2,03E+01	9,63E+00	2,25E+01	2,22E+00	2,00E-02	1,83E+00	2,79E+00	1,26E+02
	7	4,18E+00	5,70E-02	2,04E+01	1,54E+01	2,76E+01	7,00E+00	1,03E+01	2,24E+00	3,43E+00	2,17E+02
	13(Median)	4,82E+00	1,41E-01	2,05E+01	1,82E+01	3,36E+01	8,08E+00	1,25E+01	3,07E+00	3,66E+00	2,65E+02
1E+04	19	6,21E+00	3,97E-01	2,05E+01	2,24E+01	3,62E+01	8,85E+00	5,52E+01	3,20E+00	3,76E+00	4,02E+02
	25(Worst)	9,25E+00	5,61E-01	2,06E+01	2,58E+01	3,84E+01	9,99E+00	7,27E+02	3,33E+00	3,81E+00	4,27E+02
	Mean	4,97E+00	2,15E-01	2,05E+01	1,79E+01	3,19E+01	7,61E+00	8,93E+01	2,82E+00	3,56E+00	2,81E+02
	Std. Dev.	2,22E+00	1,90E-01	9,23E-02	4,46E+00	5,04E+00	1,84E+00	1,99E+02	5,01E-01	2,49E-01	9,91E+01
	1(Best)	0,00E+00	7,00E-03	2,02E+01	0,00E+00	3,01E+00	1,00E-04	0,00E+00	5,25E-01	1,02E+00	0,00E+00
	7	2,50E-02	4,40E-02	2,03E+01	9,95E-01	5,97E+00	2,02E-01	9,40E-02	9,13E-01	2,13E+00	6,30E+01
	13(Median)	4,50E-02	5,70E-02	2,03E+01	1,99E+00	7,97E+00	1,09E+00	1,00E+01	1,41E+00	2,42E+00	8,97E+01
1E+05	19	8,70E-02	9,60E-02	2,04E+01	2,99E+00	1,44E+01	1,75E+00	1,88E+01	1,72E+00	2,75E+00	1,39E+02
	25(Worst)	4,00E+00	1,30E-01	2,04E+01	3,98E+00	2,11E+01	2,01E+00	7,12E+02	1,87E+00	2,92E+00	4,20E+02
	Mean	**4,75E-01**	**6,46E-02**	**2,03E+01**	**1,99E+00**	**1,02E+01**	**1,00E+00**	**3,70E+01**	**1,32E+00**	**2,31E+00**	**1,34E+02**
	Std. Dev.	1,12E+00	3,18E-02	5,01E-02	1,18E+00	5,12E+00	7,65E-01	1,41E+02	4,45E-01	5,42E-01	1,28E+02

Table 4.3: Error values for functions f_{16} to f_{25} with dimension $D = 10$ and 100,000 evaluations

Evals.	Exec. Num.	$\mathbf{f_{16}}$	$\mathbf{f_{17}}$	$\mathbf{f_{18}}$	$\mathbf{f_{19}}$	$\mathbf{f_{20}}$	$\mathbf{f_{21}}$	$\mathbf{f_{22}}$	$\mathbf{f_{23}}$	$\mathbf{f_{24}}$	$\mathbf{f_{25}}$
	1(Best)	2,00E+02	1,69E+02	8,43E+02	8,41E+02	4,08E+02	5,66E+02	6,26E+02	6,16E+02	3,02E+02	2,79E+02
	7	2,30E+02	2,63E+02	9,32E+02	9,37E+02	5,81E+02	9,85E+02	8,23E+02	1,02E+03	4,17E+02	4,13E+02
	13(Median)	2,50E+02	2,85E+02	1,01E+03	1,01E+03	7,12E+02	1,16E+03	8,38E+02	1,20E+03	5,98E+02	6,05E+02
1E+03	19	2,66E+02	2,93E+02	1,06E+03	1,06E+03	8,33E+02	1,22E+03	8,64E+02	1,22E+03	7,49E+02	7,74E+02
	25(Worst)	2,73E+02	3,17E+02	1,08E+03	1,08E+03	9,63E+02	1,24E+03	9,44E+02	1,25E+03	9,10E+02	1,02E+03
	Mean	2,47E+02	2,71E+02	9,87E+02	9,98E+02	7,18E+02	1,05E+03	8,44E+02	1,11E+03	5,94E+02	5,96E+02
	Std. Dev.	2,08E+01	3,32E+01	7,10E+01	7,42E+01	1,61E+02	2,23E+02	5,98E+01	1,76E+02	1,96E+02	2,16E+02
	1(Worst)	1,36E+02	1,58E+02	3,00E+02	3,00E+02	2,00E+02	3,00E+02	1,00E+02	3,00E+02	2,00E+02	2,00E+02
	7	1,64E+02	1,88E+02	8,00E+02	8,00E+02	2,00E+02	3,00E+02	7,74E+02	3,00E+02	2,00E+02	2,00E+02
	13(Median)	1,71E+02	1,97E+02	8,00E+02	8,00E+02	2,00E+02	3,00E+02	7,76E+02	3,00E+02	2,00E+02	2,00E+02
1E+04	19	1,78E+02	2,05E+02	8,00E+02	9,08E+02	2,00E+02	5,00E+02	8,00E+02	8,00E+02	2,00E+02	2,00E+02
	25(Worst)	1,88E+02	2,17E+02	9,33E+02	9,49E+02	2,00E+02	9,00E+02	8,00E+02	1,05E+03	5,00E+02	5,00E+02
	Mean	1,68E+02	1,94E+02	7,05E+02	7,83E+02	2,00E+02	4,64E+02	7,37E+02	5,23E+02	2,68E+02	2,48E+02
	Std. Dev.	1,46E+01	1,59E+01	2,21E+02	1,79E+02	0,00E+00	2,12E+02	1,65E+02	2,76E+02	1,25E+02	1,12E+02
	1(Best)	8,21E+01	1,03E+02	3,00E+02	3,00E+02	2,00E+02	3,00E+02	1,00E+02	3,00E+02	2,00E+02	2,00E+02
	7	9,84E+01	1,17E+02	8,00E+02	8,00E+02	2,00E+02	3,00E+02	7,60E+02	3,00E+02	2,00E+02	2,00E+02
	13(Median)	1,06E+02	1,24E+02	8,00E+02	8,00E+02	2,00E+02	3,00E+02	7,64E+02	3,00E+02	2,00E+02	2,00E+02
1E+05	19	1,12E+02	1,29E+02	8,00E+02	9,08E+02	2,00E+02	5,00E+02	8,00E+02	8,00E+02	2,00E+02	2,00E+02
	25(Worst)	1,20E+02	1,52E+02	9,32E+02	9,46E+02	2,00E+02	9,00E+02	8,00E+02	1,05E+03	5,00E+02	5,00E+02
	Mean	**1,05E+02**	**1,25E+02**	**7,05E+02**	**7,82E+02**	**2,00E+02**	**4,64E+02**	**7,29E+02**	**5,23E+02**	**2,68E+02**	**2,48E+02**
	Std. Dev.	9,49E+00	1,05E+01	2,21E+02	1,78E+02	0,00E+00	2,12E+02	1,63E+02	2,76E+02	1,25E+02	1,12E+02

Table 4.4: Error values for functions f_6 to f_{15} with dimension $D = 30$ and 300,000 evaluations

Evals.	Exec. Num.	$\mathbf{f_6}$	$\mathbf{f_7}$	$\mathbf{f_8}$	$\mathbf{f_9}$	$\mathbf{f_{10}}$	$\mathbf{f_{11}}$	$\mathbf{f_{12}}$	$\mathbf{f_{13}}$	$\mathbf{f_{14}}$	$\mathbf{f_{15}}$
	1(Best)	7,26E+08	2,92E+01	2,10E+01	1,88E+02	2,13E+02	3,97E+01	1,09E+05	1,92E+01	1,35E+01	4,86E+02
	7	2,15E+09	3,99E+01	2,11E+01	2,23E+02	2,68E+02	4,17E+01	1,50E+05	2,36E+01	1,37E+01	5,56E+02
	13(Median)	3,18E+09	4,88E+01	2,11E+01	2,43E+02	2,79E+02	4,31E+01	1,79E+05	2,46E+01	1,38E+01	6,10E+02
3E+03	19	4,77E+09	5,89E+01	2,12E+01	2,60E+02	2,87E+02	4,35E+01	2,10E+05	2,50E+01	1,39E+01	6,49E+02
	25(Worst)	9,68E+09	6,39E+01	2,12E+01	2,66E+02	2,94E+02	4,50E+01	2,96E+05	2,59E+01	1,41E+01	7,32E+02
	Mean	3,68E+09	4,88E+01	2,11E+01	2,38E+02	2,73E+02	4,27E+01	1,85E+05	2,42E+01	1,38E+01	6,04E+02
	Std. Dev.	2,24E+09	1,05E+01	5,28E-02	2,49E+01	2,04E+01	1,45E+00	5,13E+04	1,55E+00	1,49E-01	7,09E+01
	1(Best)	4,10E+00	0,00E+00	2,09E+01	5,59E+01	1,50E+02	3,85E+01	1,04E+03	1,56E+01	1,28E+01	2,06E+02
	7	2,32E+01	1,70E-02	2,10E+01	1,15E+02	1,97E+02	4,03E+01	4,50E+03	1,70E+01	1,32E+01	2,20E+02
	13(Median)	2,59E+01	2,50E-02	2,10E+01	1,41E+02	2,08E+02	4,10E+01	8,58E+03	1,80E+01	1,34E+01	3,00E+02
3E+04	19	7,36E+01	3,80E-02	2,11E+01	1,52E+02	2,24E+02	4,14E+01	1,17E+04	1,85E+01	1,36E+01	3,30E+02
	25(Worst)	2,00E+03	5,40E-02	2,11E+01	1,60E+02	2,33E+02	4,20E+01	2,00E+04	1,92E+01	1,36E+01	4,06E+02
	Mean	1,43E+02	2,76E-02	2,10E+01	1,30E+02	2,06E+02	4,08E+01	8,59E+03	1,77E+01	1,34E+01	2,95E+02
	Std. Dev.	3,99E+02	1,36E-02	3,96E-02	3,09E+01	2,19E+01	9,25E-01	5,19E+03	1,06E+00	2,23E-01	7,34E+01
	1(Best)	0,00E+00	0,00E+00	2,08E+01	1,49E+01	5,38E+01	1,05E+01	4,20E+02	2,42E+00	1,20E+01	2,02E+02
	7	1,00E-03	1,00E-02	2,09E+01	2,29E+01	1,50E+02	1,32E+01	1,47E+03	3,97E+00	1,27E+01	2,17E+02
	13(Median)	5,90E-02	1,00E-02	2,09E+01	2,49E+01	1,74E+02	1,58E+01	2,60E+03	1,10E+01	1,28E+01	3,00E+02
3E+05	19	3,99E+00	2,00E-02	2,10E+01	2,69E+01	1,79E+02	3,18E+01	5,08E+03	1,30E+01	1,30E+01	3,29E+02
	25(Worst)	8,50E+00	3,20E-02	2,10E+01	3,38E+01	1,91E+02	3,81E+01	8,22E+03	1,58E+01	1,31E+01	4,04E+02
	Mean	**1,75E+00**	**1,34E-02**	**2,09E+01**	**2,49E+01**	**1,64E+02**	**2,06E+01**	**3,30E+03**	**9,65E+00**	**1,28E+01**	**2,90E+02**
	Std. Dev.	2,51E+00	7,95E-03	4,63E-02	4,84E+00	2,86E+01	1,06E+01	2,43E+03	4,80E+00	2,76E-01	7,64E+01

Table 4.5: Error values for functions f_{16} to f_{25} with dimension $D = 30$ and 300,000 evaluations

Evals.	Exec. Num.	$\mathbf{f_{16}}$	$\mathbf{f_{17}}$	$\mathbf{f_{18}}$	$\mathbf{f_{19}}$	$\mathbf{f_{20}}$	$\mathbf{f_{21}}$	$\mathbf{f_{22}}$	$\mathbf{f_{23}}$	$\mathbf{f_{24}}$	$\mathbf{f_{25}}$
	1(Best)	2,66E+02	3,01E+02	9,07E+02	9,05E+02	9,92E+02	5,20E+02	6,36E+02	5,22E+02	4,38E+02	4,14E+02
	7	2,94E+02	3,41E+02	9,18E+02	9,20E+02	1,01E+03	5,34E+02	6,57E+02	5,30E+02	4,94E+02	5,40E+02
	13(Median)	3,18E+02	3,72E+02	9,27E+02	9,24E+02	1,05E+03	5,51E+02	7,30E+02	5,41E+02	5,56E+02	5,88E+02
3E+03	19	3,30E+02	4,44E+02	9,35E+02	9,29E+02	1,16E+03	6,11E+02	8,79E+02	5,86E+02	5,77E+02	6,19E+02
	25(Worst)	4,29E+02	4,86E+02	9,37E+02	9,48E+02	1,27E+03	1,00E+03	9,05E+02	8,44E+02	6,92E+02	7,00E+02
	Mean	3,27E+02	3,84E+02	9,26E+02	9,25E+02	1,09E+03	6,12E+02	7,67E+02	5,91E+02	5,46E+02	5,78E+02
	Std. Dev.	4,56E+01	5,79E+01	9,44E+00	9,37E+00	1,00E+02	1,43E+02	1,09E+02	1,03E+02	6,67E+01	7,44E+01
	1(Best)	2,08E+02	2,43E+02	8,58E+02	8,59E+02	9,00E+02	5,09E+02	5,02E+02	5,09E+02	2,38E+02	2,39E+02
	7	2,21E+02	2,61E+02	8,63E+02	8,62E+02	9,00E+02	5,09E+02	5,04E+02	5,10E+02	2,42E+02	2,43E+02
	13(Median)	2,46E+02	2,74E+02	8,64E+02	8,64E+02	9,00E+02	5,10E+02	5,05E+02	5,10E+02	2,44E+02	2,45E+02
3E+04	19	2,59E+02	3,01E+02	8,65E+02	8,66E+02	9,00E+02	5,10E+02	5,50E+02	5,10E+02	2,47E+02	2,47E+02
	25(Worst)	3,37E+02	3,76E+02	8,69E+02	8,71E+02	1,18E+03	8,00E+02	5,50E+02	5,10E+02	2,49E+02	2,48E+02
	Mean	2,52E+02	2,91E+02	8,64E+02	8,64E+02	9,40E+02	5,33E+02	5,21E+02	5,10E+02	2,44E+02	2,45E+02
	Std. Dev.	3,66E+01	4,21E+01	2,75E+00	3,08E+00	9,31E+01	8,04E+01	2,25E+01	2,25E-01	3,14E+00	2,49E+00
	1(Best)	3,82E+01	6,40E+01	8,35E+02	8,27E+02	9,00E+02	5,09E+02	5,00E+02	5,09E+02	2,32E+02	2,32E+02
	7	1,73E+02	2,21E+02	8,55E+02	8,56E+02	9,00E+02	5,09E+02	5,00E+02	5,10E+02	2,34E+02	2,33E+02
	13(Median)	2,01E+02	2,27E+02	8,59E+02	8,58E+02	9,00E+02	5,10E+02	5,01E+02	5,10E+02	2,34E+02	2,34E+02
3E+05	19	2,17E+02	2,48E+02	8,62E+02	8,60E+02	9,00E+02	5,10E+02	5,50E+02	5,10E+02	2,35E+02	2,35E+02
	25(Worst)	2,93E+02	3,25E+02	8,65E+02	8,62E+02	1,18E+03	8,00E+02	5,50E+02	5,10E+02	2,35E+02	2,37E+02
	Mean	**1,87E+02**	**2,22E+02**	**8,57E+02**	**8,55E+02**	**9,40E+02**	**5,33E+02**	**5,18E+02**	**5,10E+02**	**2,34E+02**	**2,34E+02**
	Std. Dev.	7,11E+01	6,59E+01	8,08E+00	8,57E+00	9,31E+01	8,04E+01	2,43E+01	2,24E-01	7,55E-01	1,34E+00

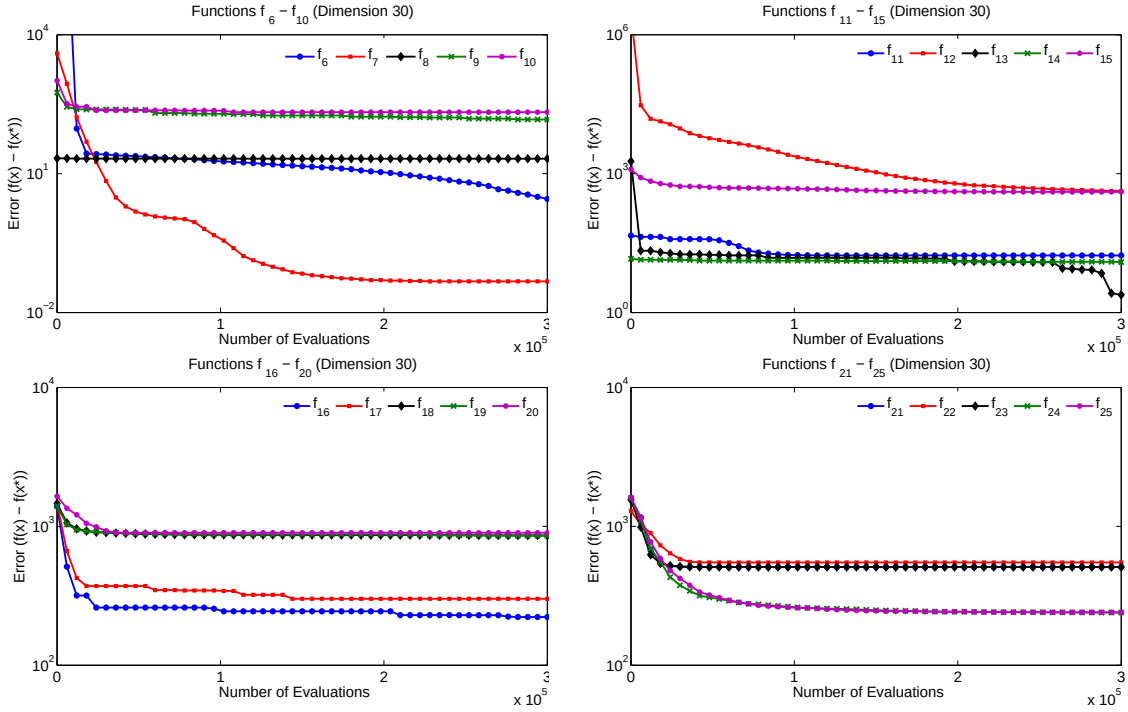


Figure 4.1: Median execution trace of DEPSO for dimension 30. The mean error value in logarithmic scale is plotted for each benchmark function

Table 4.6: Signed Rank test of DEPSO in comparison with G-CMA-ES, K-PCX and DE, for dimensions 10 and 30 and with confidence level 95% (p-value=0.05)

Algorithm	Dimension	$R+$	$R-$	p-value
G-CMA-ES	10	**111**	79	0,520
	30	103	**107**	0,940
DE	10	102	**108**	0,911
	30	82	**128**	0,391
K-PCX	10	66	**144**	0,145
	30	**111**	79	0,520

Now, we directly compare the mean values resulted from our DEPSO with the ones of CEC'05 suggested in MAEB'09 procedure. Then, in Table 4.7 there is a summary of algorithms outperformed by our proposal for each benchmarking function and dimension. Columns 3 and 5 in this table contains the relative position in which DEPSO is arranged with regards to other compared algorithms. In this way, for function f_{15}, DEPSO obtains better results than DE, K-PCX and G-CMA-ES on dimension 10 (position=1) and better results than DE and K-PCX on dimension 30 (position=2). Therefore, we can observe that for dimension 10, DEPSO reaches the best result for 6 functions and the worst one for 4 functions (symbol -). With dimension 30, DEPSO obtains the best result for 3 functions, outperforms at least two other algorithms on 6 functions, and shows the worst results only on 3 out of 20 functions.

In particular, if we examine the performance of DEPSO for each benchmark function, we can observe that from functions f_{14} to f_{25}, this algorithm obtains in general better results than for

Table 4.7: Ranking positions of DEPSO for each function with regards to compared algorithms

Function	Dimension 10		Dimension 30	
	Alg. CEC'05	#DEPSO	Alg. CEC'05	#DEPSO
f_6	DE	3	DE, K-PCX	2
f_7	DE, K-PCX	2	K-PCX	3
f_8	DE	3	DE	3
f_9	-	4	-	4
f_{10}	DE	3	-	4
f_{11}	K-PCX	3	DE, K-PC	2
f_{12}	K-PCX	3	C-MA-ES	3
f_{13}	-	4	DE	3
f_{14}	**DE, K-PCX, C-MA-ES**	**1**	**DE, K-PCX, C-MA-ES**	**1**
f_{15}	**DE, K-PCX, C-MA-ES**	**1**	DE, K-PCX	2
f_{16}	DE	3	DE	3
f_{17}	-	4	DE, C-MA-ES	2
f_{18}	K-PCX	3	DE, C-MA-ES	2
f_{19}	-	4	DE, C-MA-ES	2
f_{20}	**DE, K-PCX, C-MA-ES**	**1**	-	4
f_{21}	**DE, K-PCX, C-MA-ES**	**1**	K-PCX	3
f_{22}	C-MA-ES	3	**DE, K-PCX, C-MA-ES**	**1**
f_{23}	**DE, K-PCX, C-MA-ES**	**1**	**DE, K-PCX, C-MA-ES**	**1**
f_{24}	K-PCX	3	C-MA-ES	3
f_{25}	**DE, K-PCX, C-MA-ES**	**1**	DE	3

previous functions, f_1 to f_{13}. That is, the performance of our proposal is relatively better for composed functions than for simple ones, with regards to the three compared algorithms. We have to notice that functions f_{14} to f_{25} are in general rotated non-separable, so this fact leads us to argue that hybridizing PSO with DE differential operators provides the first algorithm with search capabilities on these kind of functions, on which, PSO initially showed a limited performance. In addition, DEPSO also shows better performance than canonical DE, since for dimension 10 the former outperformed the later on 11 functions (out of 20), and for dimension 30, DEPSO outperformed DE on 14 functions out of 20. Nevertheless, there are still three exceptions on functions: f_{17}, f_{19} and f_{20}, for which DEPSO resulted in the last position. It might be due to the intrinsic noisy nature of these three functions with narrow basin of global optimum, so let us experiment this issue thoroughly on another extensive, and completely different benchmark, with noisy and noiseless functions in next section of this chapter.

4.4 Experiments on BBOB'09

In the case of BBOB'09 test suite, the experimentation procedure has been carried out according to [HAFR09a] on the two benchmarks of: 24 noiseless functions given in [FHRA09a, HFRA09a] and 30 noisy functions given in [FHRA09b, HFRA09b]. These two sets of functions were tackled connecting the C-code of the Black-Box Optimization Benchmarking to our implementation of DEPSO. Each candidate solution was sampled uniformly in $[-5, 5]^D$, where D represents the search space dimension. The maximum number of function evaluation was set to $1000 \times D$.

Our proposal was tested performing 15 independent runs for each noiseless and noisy function and each dimension. Table 4.1 shows the parameter setting used to configure DEPSO. As previously explained, these parameters were tuned in the context of the special session of MAEB'09 for real parameter optimization [SHL$^+$05, GNAAL09] reaching results statistically similar to the best participant algorithms (G-CMA-ES and K-PCX) in that session. This parameterization was kept the same for all the experiments, and therefore the *crafting effort* [HAFR09a] is zero.

Table 4.8: Noiseless Functions ranked by number of successful trials obtained by DEPSO

Dimensions					
2	3	5	10	20	40
f1	f1	f1	f1	f5	f5
f2	f2	f2	f5	-	-
f3	f5	f5	f2	-	-
f5	f6	f7	f21	-	-
f6	f21	f21	-	-	-
f7	f7	f6	-	-	-
f21	f3	f22	-	-	-
f22	f20	-	-	-	-
f20	f22	-	-	-	-
f12	f4	-	-	-	-
f15	f17	-	-	-	-
f9	-	-	-	-	-
f17	-	-	-	-	-
f19	-	-	-	-	-

Table 4.9: Noisy Functions ranked by number of successful trials obtained by DEPSO

Dimensions					
2	3	5	10	20	40
f101	f101	f101	f101	-	-
f102	f102	f102	f102	-	-
f113	f107	f107	-	-	-
f128	f113	f113	-	-	-
f107	f128	f128	-	-	-
f103	f103	-	-	-	-
f115	f115	-	-	-	-
f125	-	-	-	-	-
f130	-	-	-	-	-
f116	-	-	-	-	-
f105	-	-	-	-	-

4.4.1 GECCO'09: Numerical Experiments on Noiseless Functions

In this section, the results are presented and some discussions are made in terms of the number of successful trials ($f_{\text{opt}} + \Delta f \leq 10^{-8}$) reached for each kind of noiseless function: separable (f1-f5), moderate (f6-f9), ill-conditioned (f10-f14), multimodal (f15-f19), and weak structure (f20-f24).

Figure 4.2 shows the Expected Running Time (ERT, ●) to reach $f_{\text{opt}} + \Delta f$ and median number of function evaluations of successful trials (+). As we can observe, our proposal obtains the highest number of successful trials in separable, moderate and weak structure noiseless functions, specifically in f1, f2, f5, f6, f7, f21, and f22 for dimensions 2, 3, and 5. We can notice that for f5 our DEPSO reaches the target optimum for all dimensions. In the scope of multimodal functions, a lower number of successful trials was found for dimension 2 and 3 in f15, f17, and f19. For ill-conditioned functions, only f12 shows successful trials (six) with dimension 2. As a summary, in Table 4.8 the functions are ranked by means of the number of successful trials that DEPSO has obtained with them (with the best ranked functions in the top).

In the analysis of the results with higher dimensions (20 and 40), the best behavior of our proposal can be observed when facing separable noiseless functions as shown in Figure 4.2. Concretely, with functions f1 and f5 where error precisions close to 10^{-8} were reached. In addition, DEPSO obtained error values lower than 10^{-7} in the weak structured function f21 with dimension 20. Concerning the remaining functions, the best achieved Δf precisions (of the median trials) obtained by our proposal are always lower than $e + 0$, although being some of them close to $e - 3$ as in f2 and f14. Only for f10, f15, and f14 our DEPSO obtained Δf precisions higher than $e + 0$.

4.4.2 GECCO'09: Numerical Experiments on Noisy Functions

In this section, the results are presented and some discussions are made in terms of the number of successful trials ($f_{\text{opt}} + \Delta f \leq 10^{-8}$) reached by DEPSO for each kind of noisy function: moderate (f101-f106), severe (f107-f121), and severe multimodal (f122-f139).

Figure 4.3 shows the Expected Running Time (ERT, ●) to reach $f_{\text{opt}} + \Delta f$ and median number of function evaluations of successful trials (+). As we can observe, our proposal obtains the highest

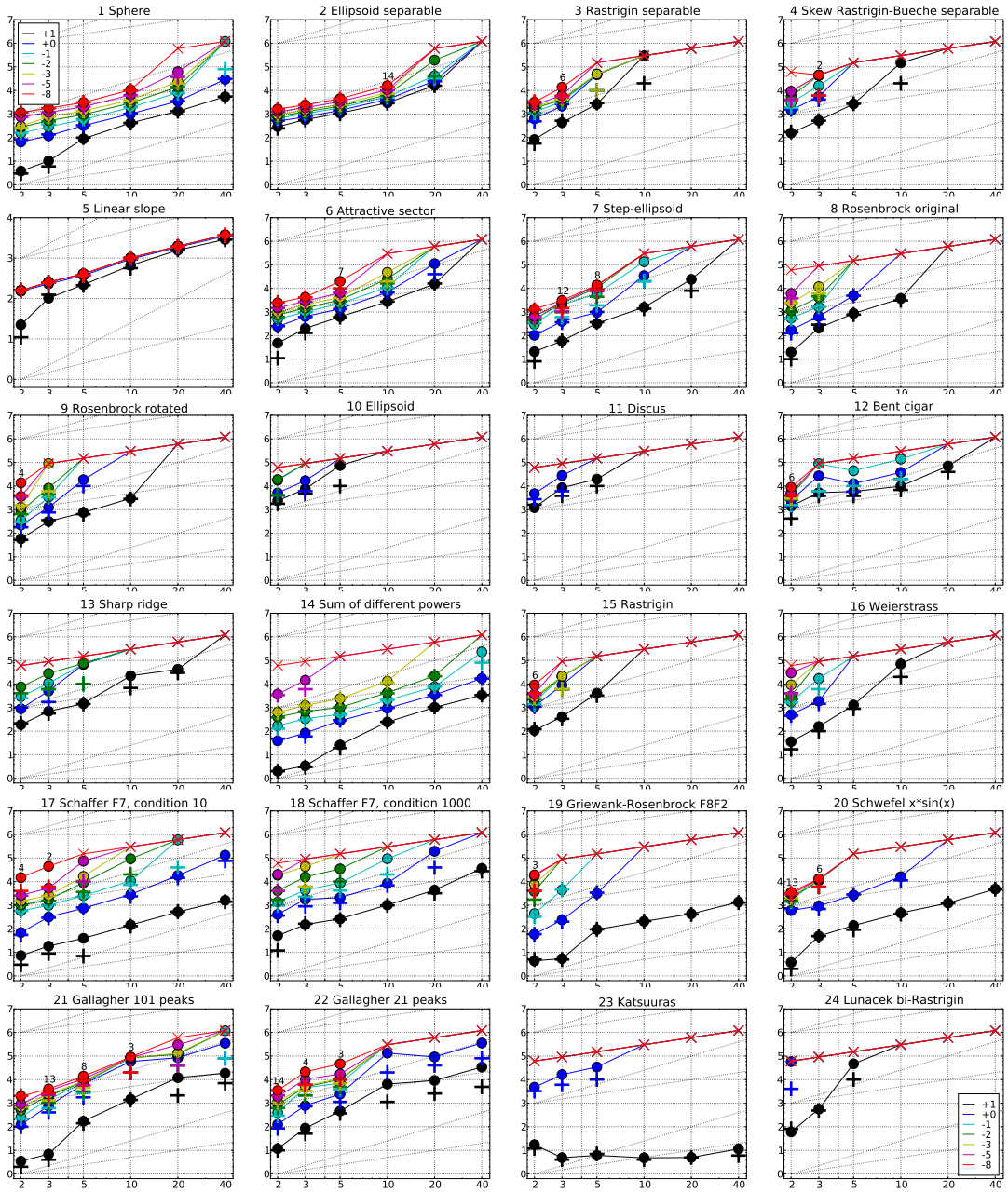


Figure 4.2: Expected Running Time (ERT, ●) to reach $f_{\text{opt}} + \Delta f$ and median number of function evaluations of successful trials (+), shown for $\Delta f = 10, 1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-5}, 10^{-8}$ (the exponent is given in the legend of f_1 and f_{24}) versus dimension in log-log presentation. The ERT(Δf) equals to #FEs(Δf) divided by the number of successful trials, where a trial is successful if $f_{\text{opt}} + \Delta f$ was surpassed during the trial. The #FEs(Δf) are the total number of function evaluations while $f_{\text{opt}} + \Delta f$ was not surpassed during the trial from all respective trials (successful and unsuccessful), and f_{opt} denotes the optimal function value. Crosses (×) indicate the total number of function evaluations #FEs($-\infty$). Numbers above ERT-symbols indicate the number of successful trials. Annotated numbers on the ordinate are decimal logarithms. Additional grid lines show linear and quadratic scaling

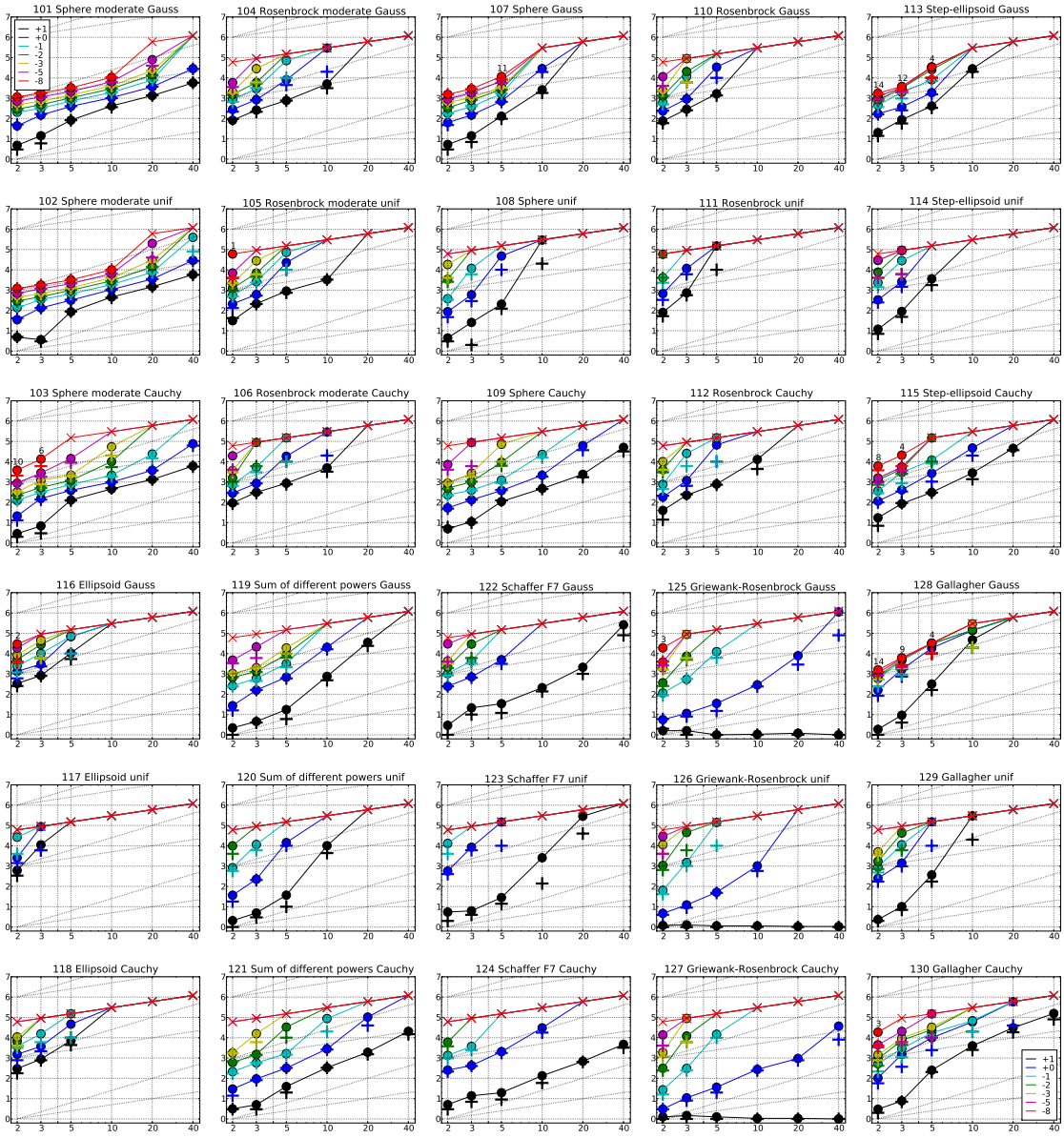


Figure 4.3: Expected Running Time (ERT, ●) to reach $f_{\text{opt}} + \Delta f$ and median number of function evaluations of successful trials (+), shown for $\Delta f = 10, 1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-5}, 10^{-8}$ (the exponent is given in the legend of f_{101} and f_{130}) versus dimension in log-log presentation. The ERT(Δf) equals to #FEs(Δf) divided by the number of successful trials, where a trial is successful if $f_{\text{opt}} + \Delta f$ was surpassed during the trial. The #FEs(Δf) are the total number of function evaluations while $f_{\text{opt}} + \Delta f$ was not surpassed during the trial from all respective trials (successful and unsuccessful), and f_{opt} denotes the optimal function value. Crosses (×) indicate the total number of function evaluations #FEs($-\infty$). Numbers above ERT-symbols indicate the number of successful trials. Annotated numbers on the ordinate are decimal logarithms. Additional grid lines show linear and quadratic scaling

number of successful trials in moderate noise functions, specifically in dimensions 2, 3, 5 and 10, in f101 and f102. With regards to severe noise functions, the target function is reached in 9 out of 15 trials for the lower dimensions. In severe noise multimodal functions, DEPSO obtains successful trials in f125 and f130 for dimension 2, and in f128 for dimensions 2, 3 and 5. As a summary, in Table 4.9 the functions are ranked by means of the number of successful trials that DEPSO has obtained with them (with the best ranked functions in the top).

For higher dimensions (20 and 40), the best behavior of our proposal can be observed when facing moderate noise functions as shown in Figure 4.3, concretely in functions f101 and f102 where error precisions slightly higher than 10^{-5} were reached. Secondly, for severe noise and severe noise multimodal functions (f122-f139, D=20), the best achieved Δf precisions (of the median trials) are always close to $e+0$, although being some of them close to $e-2$ as in f109, f125, and f127. We suspect that the relative low convergence rate in severe noise functions (for the higher dimensions) can be due to the small number of maximal function evaluations employed ($1000 \times D$).

4.5 Run Time Analysis

For the timing experiment, the same DEPSO algorithm was run on f8 of BBOB'09 until at least 30 seconds had passed (according to the `exampletiming` procedure available in BBOB 2009 [HAFR09a]). These experiments have been conducted with an Intel(R) Corel(TM)2 CPU processor with 1.66 GHz and 1GB RAM; O.S Linux Ubuntu version 8.10 using the C-code provided. The results were 1.1, 0.9, 1.3, 2.3, 3.9, and 7.1 $\times$ e-06 seconds per function evaluation in dimensions 2, 3, 5, 10, 20, and 40, respectively.

4.6 Conclusions

In this chapter we have empirically studied DEPSO, an easy to implement optimization algorithm constructed by hybridizing the Particle Swarm Optimizer with Differential Evolution operations. The experiments have been completed in the context of two Special Sessions of Real Parameter Optimization with different sets of functions: MAEB'09/CEC'05 and GECCO BBOB'09. A total number of 74 different functions of continuous optimization have been used to assess the performance of DEPSO, dealing with complex noiseless and noisy functions with dimension: 2, 3, 5, 10, 20, 30 and 40 variables. In general, the following conclusions can be highlighted:

- On MAEB'09 test set, we experimented that hybridizing PSO with DE differential operators provides the first algorithm with search capabilities on rotated and non-separable functions, on which, PSO initially showed a limited performance. In addition, DEPSO also shows better performance than canonical DE on a considerable number of functions for dimensions 10 and 30. In this sense, with regards to other compared algorithms in the state of the art, DEPSO obtains better mean ranking than DE and K-PCX, for dimension $D = 10$, and better mean ranking than DE and G-CMA-ES, for dimension $D = 30$. Nevertheless, the null hypothesis can not be rejected in any case, and therefore, we can not assure that distribution of results are different.

- On BBOB'09 noiseless functions, our proposal obtained an accurate level of coverage rate for dimensions 2, 3, 5, and 10, specifically with separable and weak structured noiseless functions. Successful trials were found for 14 out of 24 functions. Specifically for f5, our proposal obtained successful trials for all dimensions.

- On BBOB'09 test set noisy functions, DEPSO obtained an accurate level of coverage rate for dimensions 2, 3, 5, and 10, specifically with moderate noise and severe noise multimodal functions. Successful trials were found for 11 out of 30 functions.

The fact of using the same parameter setting for all functions (and dimensions), together with the relatively small number of function evaluations used ($1,000 \times D$), leads us to think that DEPSO can be easily improved for better covering noiseless functions with higher dimensions. Future experiments with different parameter settings depending of each family of noiseless functions, and performing larger number of evaluations can be made in this sense.

Chapter 5

RPSO-vm: Velocity Modulation PSO for Large Scale Optimization

5.1 Introduction

In the evaluation of the search capabilities of a given optimization algorithm the usual approach is to choose a benchmark of known problems, to perform a fixed number of function evaluations, and to compare the results against the ones of other algorithms in the state of art. However, while some real industry problems can have hundreds and thousands of variables, current benchmarks are normally adopted with less than one hundred decision variables (see CEC'05 [SHL+05] and BBOB'09 [HAFR09a] testbeds). Large scale continuous optimization have attracted more and more interest (CEC'08 [TYS+07], ISDA'09 [HL09a], and CEC'10 [TLS+10]) since they introduce a high complexity to the optimization process. Issues like the exponential increment of the solution space, as well as the change that some problems exhibit as complex search landscapes with different scales, can deteriorate quickly the performance of our optimization algorithms [SQ06]. In this way, we can study certain mechanisms that show the best performance in short scale optimization problems, which is the case of the covariance matrix in G-CMA-ES [AH05], but with an unsuitable behavior for high dimensional functions (more than 100 variables). A different performance can be observed in simple algorithms like MTS [TC08], which combines several local search strategies using a small population. MTS was the best in the special session of large scale optimization of CEC'08 [TYS+07], where functions with a thousand of variables were tackled.

All this motivates us to deeply analyze the scalable capacities of optimization algorithms. In particular, Particle Swarm Optimization (PSO) [KE01] is a very simple and effective method for continuous optimization. Nevertheless, this algorithm is characterized by an early convergence behavior, mainly produced by the overinfluenced best solution and its relative facility to fall in local optima [LQSB06, VdBE04]. For this reason, PSO usually suffers from an unsuccessful performance on large dimension problems.

In this chapter, we have incorporated two mechanisms to the Particle Swarm Optimization with the aim of enhancing its scalability [GNA11b]. First, a *velocity modulation* method is applied in the movement of particles in order to guide them within the bounded search region. Second, a *restarting* mechanism avoids the early convergence and redirects particles to promising areas in the search space. To evaluate the scalability of the resulting approach, we have followed the experimental framework proposed in the Special Issue of Soft Computing Journal on *Scalability*

of Evolutionary Algorithms and other Metaheuristics for Large Scale Continuous Optimization Problems (in URL `http://sci2s.ugr.es/eamhco/CFP.php`). We also studied the influence of both *velocity modulation* and *restarting* mechanisms to show real insights of the improvement of our proposal, called Restart PSO with Velocity Modulation (RPSO-vm), regarding the basic PSO. The results obtained will confirm that RPSO-vm is scalable in all functions of the benchmark used, as well as highly competitive in comparison with PSO and other well-known efficient optimizers.

The remaining of this chapter is organized as follows. Next section introduces our proposal RPSO-vm. Section 5.3 describes the experimentation procedure with the benchmark of functions and the parameter settings. In Section 5.4, experimental results are reported with comparisons, analyses, and discussions. Finally, concluding remarks are given in Section 5.5.

5.2 The Algorithm: RPSO-vm

Our proposal, RPSO-vm, consists in running a PSO algorithm in which we have incorporated two main ideas: *velocity modulation* and *restarting* mechanisms.

Using the *velocity modulation*, the algorithm controls that the overall movement calculated in each evolution step and for each particle position does not exceed the limits ($\mathbf{x}_{low}$, $\mathbf{x}_{upp}$) of the problem domain. First, after calculating the new velocity value ($v^t_{aux}(j)$) RPSO-vm performs a modulation procedure as showed in Algorithm 5. The velocity vector magnitude ($\widehat{\mathbf{v}}^t_i$) is then bounded, which limits the given particle to move far from the interest area. These steps are calculated in Algorithm 6 in Lines 7 and 8. Second, for each component j of the problem domain, once obtained the new velocity $v^{t+1}_i(j)$, the overall movement is calculated, also controlling that the new particle position ($x^t_{aux}(j)$) does not exceed the problem limits for each dimension component. If this happens, the new position is recalculated by subtracting the new velocity from the old particle's position (Lines 10 to 14 in Algorithm 6).

Algorithm 5 Pseudocode of $Velmod$ procedure

1: **if** $x_{low}(j) > v^t_{aux}(j)$ **then**
2: $\quad v^{t+1}_i(j) \leftarrow x^j_{low}$
3: **else if** $v^t_{aux}(j) \geq x_{upp}(j)$ **then**
4: $\quad v^{t+1}_i(j) \leftarrow x_{upp}(j)$
5: **end if**
6: **Output:** $v^{t+1}_i(j)$ /*constricted velocity*/

A second phase of RPSO-vm concerns the *restarting* strategy. Similar to other well-known algorithms like CHC [Esh91] and G-CMA-ES [AH05], our proposal is stopped whenever one stopping criterion described below is met, and a restart is launched. The decision on when to restart the algorithm is made according to two independent criteria:

1. Stop if the standard deviation of the fitness values of particles in the entire swarm is smaller than 10^{-3}. In this case, the particles are restarted by randomly initializing their positions with a probability of $1/D$ (Lines 17 to 25 in Algorithm 6).

2. Stop if the overall change in the objective function value is below 10^{-8} for $\frac{10 \cdot D}{Ss}$ generations. In this case, the particles are restarted by calculating their derivatives to the global best position $\mathbf{b}$ and dividing them into two (Lines 26 to 32 in Algorithm 6). This way, we force the particles to go to the best but avoiding the global convergence.

Algorithm 6 Pseudocode of RPSO-vm

$S \leftarrow initializeSwarm(Ss)$ /* Swarm $S(0)$*/
while $t < MAXIMUM_t$ **do**
  /****************** Particle Swarm ******************/
  for each particle position $\mathbf{x}_i(t)$ of the swarm $S(t)$ **do**
    for each variable j of the particle's position $\mathbf{x}_i^t$ **do**
      $v_{aux}^t(j) \leftarrow velocityUpdate(\omega, v_i^t(j), \varphi_1, \varphi_2, x_i^t(i), p_i^t(j), b_i^t(i))$
      $v_i^{t+1}(j) \leftarrow velmod(v_{aux}^t(j))$
      $x_{aux}^t(j) \leftarrow x_i^t(j) + v_i^{t+1}(j)$
      if $x_{low}(j) < x_{aux}^t(j) \leq x_{upp}(j)$ **then**
        $x_i^{t+1}(j) \leftarrow x_{aux}^t(j)$
      else
        $x_i^{t+1}(j) \leftarrow x_i^t(j) - v_i^{t+1}(j)$
      end if
    end for
  end for
  /******************** Restarting ********************/
  if $std(S) < 10^{-3}$ **then**
    for each particle's position $\mathbf{x}_i(t)$ of S^t (with $\mathbf{x}_i^t \neq b^t$) **do**
      for each variable j of the particle's position $\mathbf{x}_i^t$ **do**
        if $r^t(j) < 1/D$ (with $r^t(j) \in [0,1]$) **then**
          $x_i^{t+1}(j) \leftarrow x_{low}(j) + U(0,1) \cdot (x_{upp}(j) - x_{low}(j))$
        end if
      end for
    end for
  end if
  if $change(f(\mathbf{b})) < 10^{-8}$ for $\frac{10 \cdot D}{Ss}$ steps **then**
    for each particle's position $\mathbf{x}_i^t$ of S^t (with $\mathbf{x}_i^t \neq b^t$) **do**
      for each variable j of the particle's position $\mathbf{x}_i^t$ **do**
        $x_i^{t+1}(j) \leftarrow \frac{b^t(j) - x_i^t(j)}{2}$
      end for
    end for
  end if
end while
Output: b /*The best solution found so far*/

Applying the first restarting criteria, our algorithm tries to mitigate the early stagnation that basic PSO usually suffers, specially in multimodal functions. In spite of working with high inertia and/or high social influences (φ_1 and φ_2), which moves the particles to distant positions, the PSO tends to be easily trapped in unproductive regions. This drawback is specially sensitive in functions with multiple basins of local optima such as Rastrigin and its hybrids.

The second restarting criteria is based on the existence of plateaus and quite regular regions in functions like Rosenbrock, Schwefel, and their hybrids, that makes the PSO to spent a number of function evaluations (with time and computing resources) without an effective improvement. In this case, particles tend to spread them in the search space avoiding an strong influence of the global best particle. Therefore, after certain number of function evaluations without improvement, particles are moved to their derivatives with regards to the best position.

Algorithm 6 shows the complete pseudocode of RPSO-vm. First, an initialization process of all particles in the swarm S is carried out. After this, each evolution step the particle's positions

are updated following the velocity variation model of the equations previously explained (Lines 4 to 15). If stopping criterions are reached, the algorithm restarts modifying the particles, excepting the best one (Lines 17 to 32). Finally, the algorithm returns the best solution found during the whole process.

5.3 Experimental Setup

In this section, we present the experimental methodology and statistical procedure followed to evaluate and to compare our proposal. This experimentation has been defined in the scope of the Special Issue on *Scalability of Evolutionary Algorithms and other Metaheuristics for Large Scale Continuous Optimization Problems* (SOCO'10), in URL `http://sci2s.ugr.es/eamhco/CFP.php`.

We have implemented our RPSO-vm in C++ using the MALLBA library [ALGN+07], a framework of metaheuristics. The benchmark of functions was tackled including the C-code provided in this Special Issue to our implementation of RPSO-vm[1]. Following the specifications of the SOCO'10 experimental procedure, we have performed 25 independent runs of RPSO-vm for each test function and dimension. The study has been made with dimension $D = 50$, 100, 200, 500, and $1,000$ continuous variables. The measures provided are the Average, the Maximum, the Minimum, and the Median of error of the best individuals found in the 25 runs. For a given solution $\mathbf{x}$, the error measure is defined as: $f(\mathbf{x}) - f^*$, where f^* is the optimum fitness of the function. The maximum number of fitness evaluations has been stated to $5,000 \cdot D$, which constitutes the stop condition of each run.

To analyze the results we have used non-parametric [She07] tests. These tests use the mean ranking of each algorithm. We have applied them since several times the functions might not follow the conditions of normality and homoskedasticity to apply parametric tests with security [GMLH09]. In particular, we have considered the application of the Iman and Davenport's test, and Holm's test as post-hoc procedure. The former is used to know beforehand if there are statistically relevant differences among the compared algorithms. In that case, a post-hoc procedure, the Holms test, is then employed to know which algorithms are statistically worse than the reference algorithm with the best ranking.

5.3.1 Benchmark Functions

The test suite elaborated for SOCO'10 is composed by 19 functions with different properties [HLM10b]: unimodal, multimodal, separable, non-separable, shifted, and hybrid composed. Functions 1 to 6 were defined for CEC'08 [TYS+07] and functions 7 to 11 were defined for ISDA'09 [HL09a] (and shifted for SOCO'10), where the previous ones were also used. Finally, functions 12 to 19 have been created specifically for this benchmark. Table 3.2 in Chapter 3 shows their names, bounds, features and optimum values[2]. We describe here several properties of these functions that we consider interesting.

- Functions f1 and f2 are shifted unimodal, functions f3 to f6 are shifted multimodal and functions f7 to f11 are shifted unimodal.

[1] A complete package of this software is available in the new version release of MALLBA Library `http://neo.lcc.uma.es/mallba/easy-mallba/html/mallba.html`. Directory Mallba/rep/PSO/soco2010

[2] In Table 3.2 of Chapter 3, SOCO'10 functions are referred to soco*, from soco1 to soco19

- Functions f2, f3, f5, f9, and f10 are non-separable. We are interested to analyze if our proposal obtains good results in non-separable functions, since we can observe its capacity of managing correlated variables, a typical property in real world problems.
- Functions f12 to f19 are hybrid composition functions. They have been generated by composing ($\oplus$) two functions, one or both of them non-separable. For these compositions, functions f7 to f11 have been used in their non-shifted versions (NS). A composition uses a splitting mechanism to graduate the proportion ($\oplus_{proportion}$ in Table 3.2 of Chapter 3) of non-separable variables in the complete search space.

Table 5.1: Parameter setting used in RPSO-vm

Description	Parameter	Value
Swarm size	Ss	10
Inertia weight	ω	$0.0 \leftarrow 0.1$
Individual coefficient	φ_1	1.5
Social coefficient	φ_2	1.5

5.3.2 Parameter Settings

Table 5.1 shows the parameter settings used to configure our proposal, RPSO-vm. These parameters were tuned in the context of the ISDA'09 special session of real parameter optimization [HL09a] reaching results statistically similar to the best participant algorithm in that special session. These values of parameters were kept the same for all the experiments.

5.4 Analysis of Results

In this section, the results are presented and several analyses are made as follows: first, we carry out a brief analysis of the performance of our proposal in terms of the improvement obtained by both, velocity modulation and restarting mechanisms. In this sense, an additional comparison is made concerning the neighborhood topology of RPSO-vm in terms of global best versus local best guidance of particles. Second, the scalability analysis is tackled in comparison with provided results of other algorithms (DE, CHC, and G-CMA-ES) for all dimensions. Finally, we present the computational effort required in terms of average running time.

5.4.1 RPSO-vm Numerical Results

As specified in benchmarking requirements of SOCO'10 [HLM10b], we show in Tables 5.2 and 5.3 the Average, the Maximum, the Minimum, and the Median of the best error values found in 25 independent runs of our RPSO-vm, for each function and for each dimension. In this table, we have marked in bold face the average error values since they will be used for comparisons (as recommended in this testbed). Nevertheless, we can notice that median values are frequently better than average values, especially in shifted Extended_f10 (f9) and several hybrid functions (f14, f16 and f17) where the distribution of results are scattered.

Table 5.2: Maximum, Minimum, Median, and Mean Errors obtained by RPSO-vm for functions f1 to f10 and for all dimensions

D.	Value	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10
50	Max	2.84E-14	1.58E-02	1.86E+04	2.84E-14	3.20E-01	1.78E-11	1.78E-14	2.13E+03	6.77E+00	0.00E+00
	Med	2.68E-14	6.60E-03	1.90E+01	2.66E-14	7.36E-02	2.66E-13	0.00E+00	1.04E+03	3.12E-06	0.00E+00
	Min	1.87E-14	4.36E-03	5.21E-02	0.00E+00	0.00E+00	1.24E-13	0.00E+00	4.37E+02	0.00E+00	0.00E+00
	Mean	**2.62E-14**	**7.54E-03**	**1.75E+03**	**2.30E-14**	**9.53E-02**	**1.49E-12**	**1.14E-15**	**1.06E+03**	**2.94E-01**	**0.00E+00**
100	Max	2.84E-14	2.92E-01	9.65E+03	2.84E-14	3.18E-01	9.52E-11	0.00E+00	1.89E+04	2.80E+00	0.00E+00
	Med	2.80E-14	2.01E-01	1.19E+02	2.77E-14	8.80E-02	6.32E-13	0.00E+00	1.05E+04	2.58E-05	0.00E+00
	Min	1.57E-14	1.17E-01	3.57E-01	1.17E-14	1.31E-14	2.42E-13	0.00E+00	6.82E+03	5.93E-07	0.00E+00
	Mean	**2.66E-14**	**1.98E-01**	**1.42E+03**	**2.61E-14**	**1.07E-01**	**4.76E-12**	**0.00E+00**	**1.09E+04**	**1.85E-01**	**0.00E+00**
200	Max	2.84E-14	2.49E+00	6.60E+03	2.84E-14	7.74E-01	8.20E-12	0.00E+00	7.06E+04	2.51E+01	0.00E+00
	Med	2.75E-14	2.01E+00	6.80E+01	2.74E-14	1.18E-01	1.89E-12	0.00E+00	5.05E+04	1.48E-03	0.00E+00
	Min	0.00E+00	1.65E+00	8.75E-02	0.00E+00	1.35E-14	6.96E-13	0.00E+00	3.81E+04	1.35E-06	0.00E+00
	Mean	**2.53E-14**	**2.00E+00**	**1.03E+03**	**2.43E-14**	**1.73E-01**	**2.95E-12**	**0.00E+00**	**5.23E+04**	**1.67E+00**	**0.00E+00**
500	Max	2.84E-14	1.81E+01	1.47E+04	2.84E-14	6.01E-01	7.01E-12	1.13E-14	3.62E+05	2.28E+01	0.00E+00
	Med	2.82E-14	1.71E+01	1.38E+02	2.77E-14	2.52E-01	2.59E-12	0.00E+00	3.00E+05	1.81E+00	0.00E+00
	Min	1.73E-14	1.49E+01	2.70E-02	0.00E+00	1.42E-14	1.21E-12	0.00E+00	2.47E+05	1.90E-03	0.00E+00
	Mean	**2.65E-14**	**1.67E+01**	**1.13E+03**	**2.44E-14**	**2.54E-01**	**3.14E-12**	**0.00E+00**	**3.00E+05**	**4.85E+00**	**0.00E+00**
1,000	Max	2.84E-14	4.69E+01	2.07E+03	3.03E-13	8.68E-01	2.41E-11	1.55E-14	1.14E+06	3.17E+01	0.00E+00
	Med	2.82E-14	4.29E+01	1.37E+02	2.78E-14	1.12E-01	3.75E-12	0.00E+00	9.21E+05	9.84E+00	0.00E+00
	Min	2.07E-14	3.99E+01	1.85E+01	2.27E-14	1.22E-14	2.60E-12	0.00E+00	7.39E+05	6.45E-02	0.00E+00
	Mean	**2.72E-14**	**4.29E+01**	**3.21E+02**	**4.81E-14**	**2.14E-01**	**4.92E-12**	**0.00E+00**	**9.35E+05**	**1.17E+01**	**0.00E+00**

Table 5.3: Maximum, Minimum, Median, and Mean Errors obtained by RPSO-vm for functions f11 to f19 and for all dimensions

D.	Value	f11	f12	f13	f14	f15	f16	f17	f18	f19
50	Max	3.07E-01	2.14E+00	1.49E+04	1.04E+00	0.00E+00	1.05E-09	9.08E+03	1.26E+00	0.00E+00
	Med	3.84E-06	0.00E+00	1.43E+01	1.72E-14	0.00E+00	1.68E-11	5.06E+01	3.18E-09	0.00E+00
	Min	0.00E+00	0.00E+00	2.67E-02	0.00E+00	0.00E+00	2.20E-12	9.41E-01	4.60E-10	0.00E+00
	Mean	**1.68E-02**	**8.58E-02**	**6.57E+02**	**6.81E-02**	**0.00E+00**	**7.88E-11**	**8.73E+02**	**5.05E-02**	**0.00E+00**
100	Max	3.83E+00	2.30E-13	2.56E+04	1.20E+00	0.00E+00	1.11E-06	3.56E+04	1.61E+00	0.00E+00
	Med	1.84E-05	0.00E+00	1.06E+02	4.38E-14	0.00E+00	4.64E-10	1.51E+02	5.04E-08	0.00E+00
	Min	2.58E-08	0.00E+00	5.36E-02	0.00E+00	0.00E+00	5.17E-12	1.71E-01	1.46E-09	0.00E+00
	Mean	**4.61E-01**	**1.60E-14**	**2.25E+03**	**1.27E-01**	**0.00E+00**	**4.87E-08**	**1.76E+03**	**1.36E-01**	**0.00E+00**
200	Max	3.73E+00	1.16E+00	1.23E+05	9.95E-01	0.00E+00	1.29E+01	2.40E+05	3.73E+00	0.00E+00
	Med	5.47E-02	2.79E-14	1.48E+02	3.23E-12	0.00E+00	1.02E-09	2.77E+02	2.60E-08	0.00E+00
	Min	4.37E-05	0.00E+00	7.56E-01	1.50E-14	0.00E+00	5.35E-11	1.31E-01	3.33E-09	0.00E+00
	Mean	**5.66E-01**	**4.64E-02**	**1.17E+04**	**7.96E-02**	**0.00E+00**	**5.40E-01**	**2.08E+04**	**1.50E-01**	**0.00E+00**
500	Max	1.56E+01	2.98E-07	1.79E+04	1.36E+01	0.00E+00	3.04E+01	8.31E+03	1.41E+01	0.00E+00
	Med	3.95E+00	2.75E-13	7.20E+01	2.50E-11	0.00E+00	3.46E-08	2.16E+01	7.80E-01	0.00E+00
	Min	1.70E-03	0.00E+00	2.74E-01	2.05E-13	0.00E+00	6.84E-10	7.85E-01	1.44E-08	0.00E+00
	Mean	**4.88E+00**	**1.32E-08**	**1.33E+03**	**1.29E+00**	**0.00E+00**	**2.12E+00**	**5.72E+02**	**2.47E+00**	**0.00E+00**
1,000	Max	3.67E+01	1.16E-08	2.78E+04	2.60E+00	0.00E+00	7.53E+00	3.28E+04	6.52E+00	0.00E+00
	Med	9.61E+00	2.02E-12	2.27E+02	4.07E-08	0.00E+00	2.19E-06	4.02E+01	1.33E+00	0.00E+00
	Min	7.67E-02	2.98E-14	4.16E+01	5.39E-13	0.00E+00	6.27E-09	1.79E+00	2.93E-07	0.00E+00
	Mean	**1.10E+01**	**1.00E-09**	**1.93E+03**	**5.27E-01**	**0.00E+00**	**9.50E-01**	**2.82E+03**	**1.80E+00**	**0.00E+00**

Table 5.4: Mean Errors obtained by RPSO-vm, RPSO, PSO-vm, and PSO for dimension 1,000

F/D	RPSO-vm	RPSO	PSO-vm	PSO	(*lb*)RPSO-vm
f1	2.72E-14	**2.69E-14**	5.61E+06	5.55E+06	6.31E+06
f2	**4.29E+01**	4.38E+01	1.72E+02	1.72E+02	1.89E+02
f3	**3.21E+02**	1.26E+04	5.43E+12	5.54E+12	7.65E+12
f4	**4.81E-14**	3.98E-02	2.32E+04	2.32E+04	2.56E+04
f5	2.14E-01	**1.77E-01**	4.98E+04	5.05E+04	5.65E+04
f6	**4.92E-12**	5.13E-12	2.14E+01	2.14E+01	2.15E+01
f7	**0.00E+00**	2.00E-14	4.93E+03	4.98E+03	3.34E+27
f8	**9.35E+05**	9.38E+05	2.03E+07	2.53E+07	8.33E+07
f9	**1.17E+01**	1.34E+01	1.20E+04	1.20E+04	1.28E+04
f10	**0.00E+00**	**0.00E+00**	2.16E+05	2.16E+05	2.57E+05
f11	**1.10E+01**	1.33E+01	1.20E+04	1.20E+04	1.28E+04
f12	**1.00E-09**	1.15E-01	4.20E+06	4.18E+06	4.73E+06
f13	1.93E+03	**7.38E+02**	4.13E+12	4.01E+12	5.86E+12
f14	**5.27E-01**	6.28E-01	1.76E+04	1.78E+04	1.96E+04
f15	**0.00E+00**	**0.00E+00**	3.81E+04	3.67E+04	5.37E+18
f16	9.50E-01	**7.39E-01**	2.67E+06	2.70E+06	3.14E+06
f17	**2.82E+03**	1.42E+04	9.53E+11	9.33E+11	1.64E+12
f18	**1.80E+00**	3.35E+00	7.20E+03	7.20E+03	8.29E+03
f19	**0.00E+00**	**0.00E+00**	1.14E+05	1.11E+05	6.59E+12

Table 5.5: Comparison of RPSO-vm versus (lb)RPSO-vm, RPSO, PSO-vm, and PSO according to Holm's post-hoc multicompare test ($\alpha = 0.05$)

i	algorithm	z	p-value	α/i	Sig.dif?
4	(lb)RPSO-vm	7.02	2.09E-12	0.012	Yes
3	PSO	4.10	4.06E-05	0.016	Yes
2	PSO-vm	4.10	4.06E-05	0.025	Yes
1	RPSO	0.41	6.81E-01	0.050	No

A first analysis consists of studying the improvement obtained by RPSO-vm with regards to basic PSO algorithm. Table 5.4 shows the mean errors (in 25 runs) obtained by RPSO-vm in comparison with the ones of PSO only with restarting (RPSO), PSO only with velocity modulation (PSO-vm), and the basic Canonical PSO. Additionally, we have included to this comparison the standard version of PSO (SPSO 2007) consisting on the *lbest* PSO. This version uses a variable random topology for selecting the best neighbor (p) for each particle [GKS$^+$09]. The resulting algorithm *(lb)*RPSO-vm incorporates both, modulation velocity and restart mechanisms in order to obtain an as fair as possible comparison. For this specific analysis, we have only focused on 1,000 variables dimension, since it allows the most interesting analysis.

As we can see in Table 5.4, RPSO-vm obtains the higher number of best error values (15 out of 19 in bold), and followed by RPSO. The other versions are clearly worse than the formers. In addition, after applying Iman-Davenport test to see whether there are significant differences between them, we obtained a test value of 166.07 with a critical value of 3.77 (with $\alpha = 0.05$), which proves that there is an evident improvement of RPSO-vm over PSO.

More precisely, Table 5.5 contains the results of a multicomparison Holm's test where we can see that RPSO-vm is statistically better than all PSO versions, excepting RPSO. In this case, RPSO-vm obtained a better ranking than RPSO but without significant differences. Therefore, the main consequence is that velocity modulation (PSO-vm) can improve the performance of basic PSO, although it is in the case of PSO with restarting method (RPSO) where a significant improvement is obtained. In the case of *(lb)*RPSO-vm, we suspect that the fact of using the same parameter setting specifically fine-tuned for RPSO-vm (global best) could lead this version of PSO to perform inadequately in our experiments. These preliminary results lead us to definitively use both, the velocity modulation and the restarting method to design our RPSO-vm for large scale optimization.

Table 5.6: Results of the Iman-Davenport's (I.D.) test of RPSO-vm and all compared algorithms for each dimension ($\alpha = 0.05$)

Dimension	I.D. value	Critical value	Sig. differences?
50	13.80	2.53	Yes
100	13.43	2.53	Yes
200	12.76	2.53	Yes
500	14.37	2.53	Yes
1000	30.04	2.84	Yes

Table 5.7: Comparison of DE versus RPSO-vm, CHC, and G-CMA-ES according to Holm's multicompare test ($\alpha = 0.05$)

DIM	i	algorithm	z	$p - value$	α/i	Sig.dif?
50	3	CHC	4.77	1.79E-06	0.016	Yes
	2	G-CMA-ES	2.95	3.14E-03	0.025	Yes
	1	**RPSO-vm**	1.57	1.16E-01	0.050	**No**
100	3	CHC	4.71	2.45E-06	0.016	Yes
	2	G-CMA-ES	2.89	3.85E-03	0.025	Yes
	1	**RPSO-vm**	1.44	1.48E-01	0.050	**No**
200	3	CHC	4.649	3.33E-06	0.016	Yes
	2	G-CMA-ES	2.76	5.70E-03	0.025	Yes
	1	**RPSO-vm**	1.38	1.66E-01	0.050	**No**
500	3	CHC	4.52	6.07E-06	0.016	Yes
	2	G-CMA-ES	3.70	2.09E-04	0.025	Yes
	1	**RPSO-vm**	1.57	1.16E-01	0.050	**No**
1000	2	CHC	4.62	3.77E-06	0.025	Yes
	1	**RPSO-vm**	0.97	3.30E-01	0.050	**No**

5.4.2 Scalability Analysis

This section is focused on analyzing the capability of our RPSO-vm to scale with the dimension of the search space of each function. As proposed in SOCO'10[HLM10b], the scalability study have been made in comparison with other well-known algorithms in the state of the art. These algorithms are a version of DE (DE/1/exp) [PSL05], CHC [Esh91], and G-CMA-ES [AH05]. The descriptions and the parameter settings of these algorithms can be found in [HLM10a]. Therefore, we first analyze the results of RPSO-vm dimension by dimension, and secondly, we made a brief study from a general point of view of the scalability behavior of RPSO-vm regarding several selected functions (f2, f9, f14, and f19) of the SOCO'10 benchmark.

An initial study with all these results consists of applying an Iman-Davenport test to see if there exist significant differences between them for all considered dimensions. Table 5.6 shows the results of this test, where we can effectively notice that there are statistical differences in compared results. In fact, for almost all cases the test values (I.D. value) increase with the dimension, which means that there are higher differences between compared algorithms in large scale (500 and 1,000) than in small dimensions (50, 100 and 200). Hence, we can known beforehand that there is an algorithm with poor scalability behavior, at least.

Following this general point of view, Table 5.7 shows the results of applying a post-hoc multicomparison Holm's test to all mean fitness values obtained by each algorithm for each dimension. We must notice that G-CMA-ES could not obtained any result for dimension 1,000 (Table 5.12), hence it has not been considered for comparisons regarding the largest scale. The main observation we can draw from Table 5.7 is that there are two algorithms: DE, and RPSO-vm that clearly show a better average distribution than the remaining ones. In addition, these ranks are kept for all dimensions. Concretely, DE reached the best rank and for this reason it has been considered as the control algorithm for this statistical test. Nevertheless, our RPSO-vm is the only algorithm that

Table 5.8: Mean Errors obtained by DE, CHC, G-CMA-ES, and RPSO-vm for dimension 50

f/Alg.	DE	CHC	G-CMA-ES	RPSO-vm
f1	**0.00E+00**	1.67E-11	0.00E+00	2.62E-14
f2	3.60E-01	6.19E+01	**2.75E-11**	7.54E-03
f3	2.89E+01	1.25E+06	**7.97E-01**	1.75E+03
f4	3.98E-02	7.43E+01	1.05E+02	**2.30E-14**
f5	**0.00E+00**	1.67E-03	2.96E-04	9.53E-02
f6	**1.43E-13**	6.15E-07	2.09E+01	1.49E-12
f7	**0.00E+00**	2.66E-09	1.01E-10	**0.00E+00**
f8	3.44E+00	2.24E+02	**0.00E+00**	1.06E+03
f9	2.73E+02	3.10E+02	1.66E+01	**2.94E-01**
f10	**0.00E+00**	7.30E+00	6.81E+00	**0.00E+00**
f11	**6.23E-05**	2.16E+00	3.01E+01	1.68E-02
f12	**5.35E-13**	9.57E-01	1.88E+02	8.58E-02
f13	**2.45E+01**	2.08E+06	1.97E+02	6.57E+02
f14	**4.16E-08**	6.17E+01	1.09E+02	6.81E-02
f15	**0.00E+00**	3.98E-01	9.79E-04	**0.00E+00**
f16	1.56E-09	2.95E-09	4.27E+02	**7.88E-11**
f17	**7.98E-01**	2.26E+04	6.89E+02	8.73E+02
f18	**1.22E-04**	1.58E+01	1.31E+02	5.05E-02
f19	**0.00E+00**	3.59E+02	4.76E+00	**0.00E+00**

Table 5.9: Mean Errors obtained by DE, CHC, G-CMA-ES, and RPSO-vm for dimension 100

f/Alg.	DE	CHC	G-CMA-ES	RPSO-vm
f1	**0.00E+00**	3.56E-11	0.00E+00	2.66E-14
f2	4.45E+00	8.58E+01	**1.51E-10**	1.98E-01
f3	8.01E+01	4.19E+06	**3.88E+00**	1.42E+03
f4	7.96E-02	2.19E+02	2.50E+02	**2.61E-14**
f5	**0.00E+00**	3.83E-03	1.58E-03	1.07E-01
f6	**3.10E-13**	4.10E-07	2.12E+01	4.76E-12
f7	**0.00E+00**	1.40E-02	4.22E-04	**0.00E+00**
f8	3.69E+02	1.69E+03	**0.00E+00**	1.09E+04
f9	5.06E+02	5.86E+02	1.02E+02	**1.85E-01**
f10	**0.00E+00**	3.30E+01	1.66E+01	**0.00E+00**
f11	**1.28E-04**	7.32E+01	1.64E+02	4.61E-01
f12	5.99E-11	1.03E+01	4.17E+02	**1.60E-14**
f13	**6.17E+01**	2.70E+06	4.21E+02	2.25E+03
f14	**4.79E-02**	1.66E+02	2.55E+02	1.27E-01
f15	**0.00E+00**	8.13E+00	6.30E-01	**0.00E+00**
f16	**3.58E-09**	2.23E+01	8.59E+02	4.87E-08
f17	**1.23E+01**	1.47E+05	1.51E+03	1.76E+03
f18	**2.98E-04**	7.00E+01	3.07E+02	1.36E-01
f19	**0.00E+00**	5.45E+02	2.02E+01	**0.00E+00**

does not show significant statistical differences (Sig.dif) with regards to DE (control), and presenting the lowest difference precisely in the largest dimension (1,000 variables). This is an important indicator that confirms us the successful performance of our proposal in terms of scalability.

The following Tables 5.8, 5.9, 5.10, 5.11 and 5.12 contain the results of all the compared algorithms for dimensions: 50, 100, 200, 500 and 1,000, respectively. The last column in these tables shows the results of our RPSO-vm, indicating in bold face such values for which the mean error is the best found for each comparison.

Dimension 50. Table 5.8 shows the mean errors (of 25 runs) obtained by DE, CHC, G-CMA-ES, and RPSO-vm for dimension 50. In this case, DE obtained the best results in 13 functions, 7 of them in hybrid composition functions. RPSO-vm obtained the best results in 7 functions, and G-CMA-ES obtained the best results in 3 functions.

The Holm's test with $\alpha = 0.05$ (Table 5.7) showed that DE, the algorithm with best ranking is statistically better than all algorithms, excepting RPSO-vm with p-value$= 0.05$.

Table 5.10: Mean Errors obtained by DE, CHC, G-CMA-ES, and RPSO-vm for dimension 200

f/Alg.	DE	CHC	G-CMA-ES	RPSO-vm
f1	**0.00E+00**	8.34E-01	0.00E+00	2.53E-14
f2	1.92E+01	1.03E+02	**1.16E-09**	2.00E+00
f3	1.78E+02	2.01E+07	**8.91E+01**	1.03E+03
f4	1.27E-01	5.40E+02	6.48E+02	**2.43E-14**
f5	**0.00E+00**	8.76E-03	**0.00E+00**	1.73E-01
f6	**6.54E-13**	1.23E+00	2.14E+01	2.95E-12
f7	**0.00E+00**	2.59E-01	1.17E-01	**0.00E+00**
f8	5.53E+03	9.38E+03	**0.00E+00**	5.23E+04
f9	1.01E+03	1.19E+03	3.75E+02	**1.67E+00**
f10	**0.00E+00**	7.13E+01	4.43E+01	**0.00E+00**
f11	**2.62E-04**	3.85E+02	8.03E+02	5.66E-01
f12	**9.76E-10**	7.44E+01	9.06E+02	4.64E-02
f13	**1.36E+02**	5.75E+06	9.43E+02	1.17E+04
f14	1.38E-01	4.29E+02	6.09E+02	**7.96E-02**
f15	**0.00E+00**	2.14E+01	1.75E+00	**0.00E+00**
f16	**7.46E-09**	1.60E+02	1.92E+03	5.40E-01
f17	**3.70E+01**	1.75E+05	3.36E+03	2.08E+04
f18	**4.73E-04**	2.12E+02	6.89E+02	1.50E-01
f19	**0.00E+00**	2.06E+03	7.52E+02	**0.00E+00**

Table 5.11: Mean Errors obtained by DE, CHC, G-CMA-ES, and RPSO-vm for dimension 500

f/Alg.	DE	CHC	G-CMA-ES	RPSO-vm
f1	**0.00E+00**	2.84E-12	0.00E+00	2.65E-14
f2	5.35E+01	1.29E+02	**3.48E-04**	1.67E+01
f3	4.76E+02	1.14E+06	**3.58E+02**	1.13E+03
f4	3.20E-01	1.91E+03	2.10E+03	**2.44E-14**
f5	**0.00E+00**	6.98E-03	2.96E-04	2.54E-01
f6	**1.65E-12**	5.16E+00	2.15E+01	3.14E-12
f7	**0.00E+00**	1.27E-01	7.21E+153	**0.00E+00**
f8	6.09E+04	7.22E+04	**2.36E-06**	3.00E+05
f9	2.52E+03	3.00E+03	1.74E+03	**4.85E+00**
f10	0.00E+00	1.86E+02	1.27E+02	**0.00E+00**
f11	**6.76E-04**	1.81E+03	4.16E+03	4.88E+00
f12	**7.07E-09**	4.48E+02	2.58E+03	1.32E-08
f13	**3.59E+02**	3.22E+07	2.87E+03	1.33E+03
f14	**1.35E-01**	1.46E+03	1.95E+03	1.29E+00
f15	**0.00E+00**	6.01E+01	2.82E+262	**0.00E+00**
f16	**2.04E-08**	9.55E+02	5.45E+03	2.12E+00
f17	**1.11E+02**	8.40E+05	9.59E+03	5.72E+02
f18	**1.22E-03**	7.32E+02	2.05E+03	2.47E+00
f19	**0.00E+00**	1.76E+03	2.44E+06	**0.00E+00**

Dimension 100. In this case, in spite of reaching DE the best mean error in more functions than RPSO-vm and G-CMA-ES (see Table 5.9), both Friedman's and Holm's test showed similar results to dimension 50. That is, RPSO-vm and DE are statistically similar to themselves, although better than CHC, and G-CMA-ES.

Dimension 200. As shown in Table 5.10, the results are quite similar to the previous ones of dimension 100. In fact, the same algorithms (RPSO-vm, DE, and G-CMA-ES) obtained the best mean errors practically in the same functions. The Holm's test also confirms that DE is statistically similar to RPSO-vm (p-value$= 0.05$), and better than the rest of algorithms.

Dimension 500. Table 5.11 contains the mean errors of all algorithms in dimension 500. We can see the set functions for which RPSO-vm always obtained the best mean fitness: f4, f7, f9, f10, f15, and f19. In particular Shifted Schwefel 2.22 (f7) and its hybrids (f15 and f19) are optimized for all

Table 5.12: Mean Errors obtained by DE, CHC, and RPSO-vm for dimension 1000

f/Alg.	DE	CHC	RPSO-vm
f1	**0.00E+00**	1.36E-11	2.72E-14
f2	8.46E+01	1.44E+02	**4.29E+01**
f3	9.69E+02	8.75E+03	**3.21E+02**
f4	1.44E+00	4.76E+03	**4.81E-14**
f5	**0.00E+00**	7.02E-03	2.14E-01
f6	**3.29E-12**	1.38E+01	4.92E-12
f7	**0.00E+00**	3.52E-01	**0.00E+00**
f8	**2.46E+05**	3.11E+05	9.35E+05
f9	5.13E+03	6.11E+03	**1.17E+01**
f10	**0.00E+00**	3.83E+02	**0.00E+00**
f11	**1.35E-03**	4.82E+03	1.10E+01
f12	1.68E-08	1.05E+03	**1.00E-09**
f13	**7.30E+02**	6.66E+07	1.93E+03
f14	6.90E-01	3.62E+03	**5.27E-01**
f15	**0.00E+00**	8.37E+01	**0.00E+00**
f16	**4.18E-08**	2.32E+03	9.50E-01
f17	**2.36E+02**	2.04E+07	2.82E+03
f18	**2.37E-03**	1.72E+03	1.80E+00
f19	**0.00E+00**	4.20E+03	**0.00E+00**

dimensions. These functions are unimodal separable (f7) and unimodal non-separable (f9, f10, f15, and 19) which could led us to think that RPSO-vm only has successful performance with unimodal functions. Nevertheless, we can easily check that our proposal also obtained the best results for f4 (multimodal), and for all dimensions. In addition, RPSO-vm obtained the second best mean fitness for the remaining of hybrid composition functions (dimension 500), practically all of them characterized as multimodal. Statistically, the Holm's test confirms our initial hypothesis since it showed (Table 5.7) that the results of RPSO-vm are not significantly different to the ones of DE (p-value= 0.05), and they are statistically better than the results of the rest of algorithms.

Dimension 1000. For the largest scale, Table 5.12 shows that RPSO-vm obtained the best results for 10 out of 19 functions. As aforementioned, G-CMA-ES did not obtained any value for dimension 1,000. Regarding dimension 500, the set of functions for which RPSO-vm obtained the best mean fitness has been increased with f2, f3, f12, and f14, having these functions different properties of modality and separability. As happened in all dimensions, in spite of having DE the best average ranking, the Holm's test (Table 5.7) showed RPSO-vm is statistically similar to DE. In comparison with CHC, our proposal is statistically the best algorithm.

From a graphical point of view, Figure 5.1 illustrates the tendency of results of compared algorithms and RPSO-vm for functions f2, f9, f14, and f19 through the different dimensions. We have chosen these functions since they showed a representative behavior in terms of scalability. Therefore, we can observe in this figure that the performance of all algorithms deteriorates in higher dimensions. Nevertheless, this degradation is slight in almost all cases, and even nonexistent in others, as it happened in functions f2 and f19 for algorithms RPSO-vm and DE. A different and anomalous behavior is observed in G-CMA-ES for functions f2 and f19, where it diminishes quickly. We suspect that the use of the covariance matrix mechanism of G-CMA-ES is unsuitable for large dimensions due to the great amount of resources it requires [HMK03, KL07].

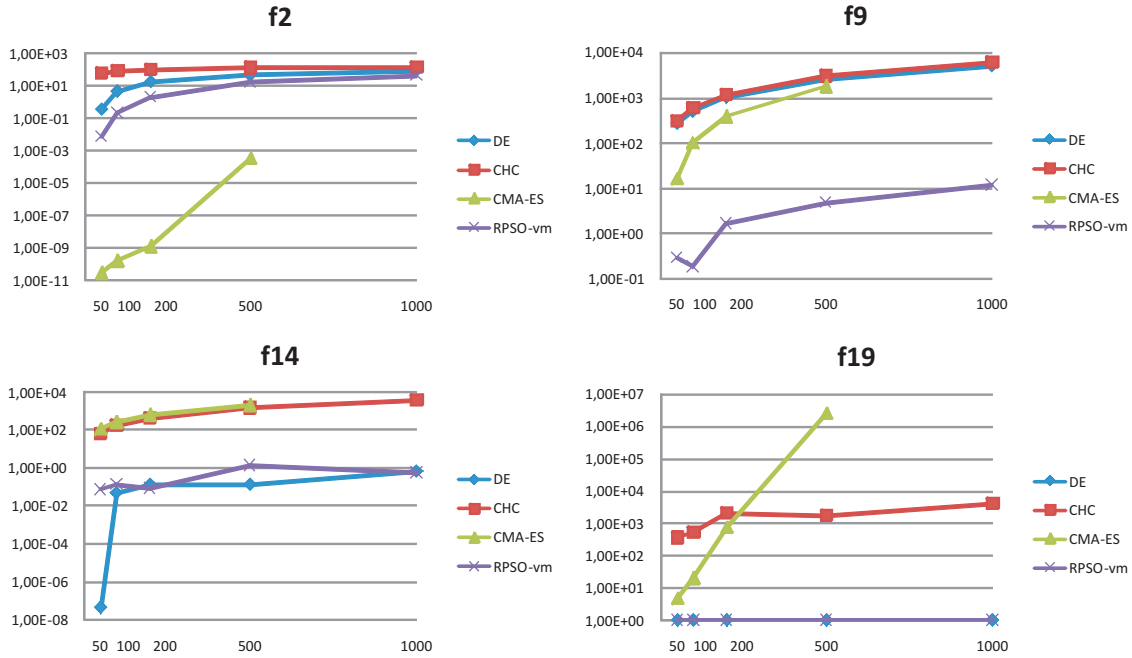


Figure 5.1: Scalable results of DE, CHC, G-CMA-ES, and RPSO-vm for functions f2, f9, f14, and f19. Y axis shows the results in logarithmic scale. X axis shows the problem dimensions

5.4.3 Computational Effort

Finally, we present in this section some remarks about the computational effort. To execute these experiments, we have used the computers of the laboratories of the Departament of Computer Science of the University of Málaga (Spain). Most of them are equipped with dual core processors, 1GB RAM, and Linux S.O., having into account that there are more than 180 computers, meaning that up to 360 cores have been available. To run all the programs, we have used the Condor [TTL05] middleware that acts as a distributed task scheduler (each task dealing with one independent run of RPSO-vm).

In Table 5.13, we present the average running time (in seconds) in which RPSO-vm has found the best mean error for all functions and for all dimensions. As expected, the running time increases with the number of variables, specially in non-separable functions. Specifically, f1 (Shifted Sphere) required the lowest time to be optimized for all dimensions, and f17 (Hybrid NS f9⊕f3) toke the longest time. In general, hybrid composition functions required more time to reach their best value than simple functions and the absolute values are low/practical, even for high dimensions.

In this sense, an interesting observation consists in comparing the increment of both, the processing time and the optimum mean error found, through the different scales of the search space. In this way, we can obtain insights about the computational effort required with regards to the quality of solutions obtained. Figure 5.2 shows a representative case observed for function f2, where the increment of the processing time as well as the mean error is practically linear. If we take into account that the search space grows exponentially with the dimension $[x_{low}, x_{upp}]^D$ in all functions, we can claim that our proposal scales successfully. Concerning the quality of solutions,

Table 5.13: Average running time (ART) in seconds of the 25 runs of RPSO-vm for all functions and for all dimensions

ART/D	50	100	200	500	1000
f1	7.91E-01	3.33E+00	1.36E+01	8.42E+01	3.60E+02
f2	2.72E+00	9.67E+00	4.10E+01	2.55E+02	9.37E+02
f3	6.30E+00	2.66E+01	9.85E+01	5.71E+02	2.50E+03
f4	2.82E+00	1.26E+01	5.50E+01	3.81E+02	1.52E+03
f5	2.19E+00	8.06E+00	3.82E+01	2.24E+02	8.60E+02
f6	4.82E+00	1.65E+01	7.43E+01	4.14E+02	1.76E+03
f7	2.42E+00	8.62E+00	3.58E+01	2.11E+02	8.72E+02
f8	2.27E+00	9.49E+00	3.41E+01	2.24E+02	8.02E+02
f9	9.05E+00	3.28E+01	1.32E+02	1.11E+03	3.19E+03
f10	4.14E+00	1.79E+01	6.40E+01	4.12E+02	1.74E+03
f11	9.23E+00	3.61E+01	1.43E+02	9.64E+02	3.92E+03
f12	4.17E+00	1.59E+01	6.63E+01	4.04E+02	1.33E+03
f13	7.44E+00	2.76E+01	9.97E+01	7.62E+02	3.03E+03
f14	5.91E+00	2.50E+01	9.30E+01	5.34E+02	2.20E+03
f15	2.78E+00	1.16E+01	4.43E+01	2.67E+02	8.90E+02
f16*	5.49E+00	2.20E+01	9.45E+01	6.09E+02	2.25E+03
f17*	8.48E+00	3.13E+01	1.41E+02	7.00E+02	3.24E+03
f18*	7.18E+00	3.05E+01	1.23E+02	7.47E+02	2.93E+03
f19*	4.01E+00	1.38E+01	6.15E+01	3.61E+02	1.43E+03

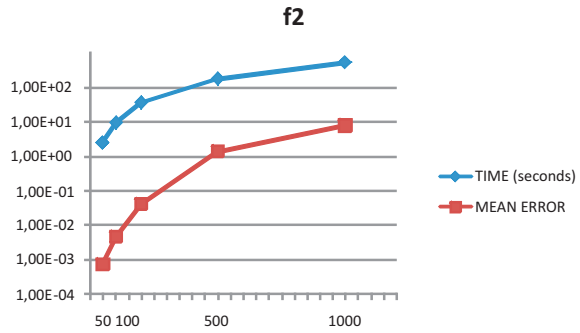


Figure 5.2: Differences in times versus mean error magnitudes of f2 for each dimension. The Y axis contains values in logarithmic scale and the X axis contains dimensions

the deterioration that the mean error suffers is higher in comparison with the processing time. Specifically, from dimension 200 to 500, the mean error increases in 2 orders of magnitude while the time required takes less than 1 order of magnitude. Curiously, the difference in the mean error between 500 and 1000 dimensions is not bigger than one order of magnitude, which leads us to suspect that our proposal performs relatively better in larger dimensions than in smaller ones.

5.5 Conclusions

In this chapter, we have incorporated both *velocity modulation* and *restarting* mechanisms to the Particle Swarm Optimization with the aim of enhancing its scalability. Our work hypothesis was that these two new mechanisms can help the PSO to avoid the early convergence and redirects the particles to promising areas in the search space. The experimentation phase has been carried out in the scope of the SOCO'10 benchmark suite to test the ability of being scalable. The results obtained show that our proposal is scalable in all functions of the benchmark used, as well as highly competitive with regards to other compared optimizers. In concrete, we can remark the following:

- the new proposal, called Restarting PSO with Velocity Modulation (RPSO-vm), outperforms the basic PSO, as well as PSO with each new mechanism separately, for all dimensions. Additionally, the RPSO-vm algorithm with *global best* neighborhood topology outperforms *(lb)*RPSO-vm: they two are the same algorithm but one has a global best (gbest) topology while the other has a variable neighborhood (lbest) topology.

- RPSO-vm shows a competitive performance in terms of its scalability. In fact, it is the second best algorithm for all dimensions and statistically similar to the best one in comparison with reference algorithms (in SOCO'10): DE, CHC, and G-CMA-ES, which are well-known optimizers traditionally used for continuous optimization and showing an excellent performance in other benchmarks (CEC'05, CEC'08, BBOB'09, etc.).

- RPSO-vm obtained the best results for functions f4, f7, f9, f10, f15, and f19 for all the dimensions. These functions are all shifted and they have different properties of modality, separability and composition. For the largest dimension (1000), the set of functions in which our algorithm obtained the best results is increased with f2, f3, f12, and f14.

- In terms of the computational effort, the running time increases with the number of variables, specially in non-separable and hybrid composition functions. Additionally, we observed that from dimension 200 to 500 the mean error increased in 2 orders of magnitude while the time required takes less than one order of magnitude. The difference in the mean error between 500 and 1,000 dimensions is not bigger than one order of magnitude. This leads us to suspect that our proposal performs relatively better in larger dimensions than in smaller ones.

In general, we can conclude that modifying the Particle Swarm Optimization algorithm, we have developed a new version (RPSO-vm) that is able to reach a highly accurate performance even in large scale environments. In the light of these results, we are encouraged to follow betting on PSO based algorithms in future works.

Chapter 6

SMPSO: Speed Modulation PSO for Multi-objective Optimization

6.1 Introduction

The relative simplicity and competitive performance of the Particle Swam Optimization [KE95] algorithm as a single-objective optimizer have favored the use of this bio-inspired technique when dealing with many real-word optimization problems [RSC06]. A considerable number of these optimization problems require to optimize more than one conflicting objectives at the same time. These trends, along with the fact that PSO is a population-based metaheuristic, have made it a natural candidate to be extended for multi-objective optimization. Since the first proposed Multi-Objective Particle Swarm Optimizer (MOPSO) developed by Moore and Chapman in 1999 [MC99], more than thirty different MOPSOs have been reported in the specialized literature. Reyes and Coello [RSC06] carried out a survey of the existing MOPSOs, providing a complete taxonomy of such algorithms. In that work, authors considered as the main features of all existing MOPSOs the following ones: the existence of an external archive of non-dominated solutions, the selection strategy of non-dominated solutions as leaders for guiding the swarm, the neighborhood topology, and the existence or not of a mutation operator.

In this chapter, we are interested in analyzing in practice six representative state-of-the-art MOPSOs in order to provide hints about their search capabilities. Five of them were selected from Reyes and Coello's survey, namely: NSPSO [Li03], SigmaMOPSO [MT03], OMOPSO [RSC05], AMOPSO [TC04], and MOPSOpd [ABEF05]. Recently, Huang et al. [HSL06] integrated a Pareto dominance concept into the Comprehensible Learning PSO (MOCLPSO), then using its learning strategy to make the particles have different learning exemplars for different dimensions.

With the aim of assessing the performance of these algorithms, we have used three benchmarks of multi-objective functions covering a broad range of problems with different features (concave, convex, disconnected, deceptive, etc.). These benchmarks include the test suites Zitzler-Deb-Thiele (ZDT) [ZDT00], the Deb-Thiele-Laumanns-Zitzler (DTLZ) problem family [DTLZ05], and the Walking-Fish-Group (WFG) test problems [HHBW06]. The experimental methodology we have followed consists in computing a pre-fixed number of function evaluations and then comparing the obtained results by considering three different quality indicators: additive unary epsilon [KTZ06], spread [DPAM02], and hypervolume [ZT99]. The results of our study reveal that many MOPSOs have difficulties when facing some multi frontal problems. We analyze this weakness and propose a

Algorithm 7 Pseudocode of a general MOPSO

1: $S \leftarrow$initializeSwarm(Ss)
2: $A \leftarrow$initializeLeadersArchive()
3: determineLeadersQuality(A)
4: **while** $t < MAXIMUM_t$ **do**
5: **for** each particle i in S **do**
6: $\mathbf{b}^t \leftarrow$ selectLeader(A^t)
7: $\mathbf{v}_i^{t+1} \leftarrow$updateVelocity($\omega, \mathbf{v}_i^t, \mathbf{x}_i^t, \varphi_1, \mathbf{p}_i^t, \varphi_2, \mathbf{b}^t$) //Equations 3.2 or 3.3
8: $\mathbf{x}_i^{t+1} \leftarrow$updatePosition($\mathbf{x}_i^t, \mathbf{v}_i^{t+1}$) //Equation 3.1
9: $\mathbf{x}_i^{t+1} \leftarrow$ mutation($\mathbf{x}_i^{t+1}$)
10: evaluate($\mathbf{x}_i^{t+1}$)
11: $\mathbf{p}_i^{t+1} \leftarrow$update($\mathbf{p}_i^t$)
12: **end for**
13: $A^{t+1} \leftarrow$updateLeadersArchive(A^t)
14: determineLeadersQuality(A^{t+1})
15: **end while**

new algorithm, called SMPSO, which incorporates a velocity constraint mechanism. We will show that SMPSO shows a promising behavior on those problems where the other algorithms fail.

The remainder of this chapter is organized as follows. Section 6.2 includes basic background about MOPSO algorithms. In Section 6.3, we briefly review the studied approaches focusing on their main features. Section 6.4 is devoted to the experimentation, including the parameter setting and the methodology adopted in the statistical tests. In Section 6.5, we analyze the obtained results regarding the three quality indicators indicated before. The results are discussed in Section 6.6, where a new MOPSO based on a constraint velocity mechanism is introduced. Finally, Section 6.7 contains the conclusions and some possible paths for future work.

6.2 The Basic MOPSO

To apply a PSO algorithm in multi-objective optimization the previous scheme has to be modified to cope with the fact that the solution of a problem with multiple objectives is not a single one but a set of non-dominated solutions. Issues that have to be considered are [RSC06]:

1. How to select the set of particles to be used as leaders?
2. How to keep the non-dominated solutions found during the search?
3. How to maintain diversity in the swarm in order to avoid convergence to a single solution?

The pseudo-code of a general MOPSO is included in Algorithm 13. After initializing the swarm (Line 1), the typical approach is to use an external archive to store the leaders, which are taken from the non-dominated particles in the swarm. After initializing the leaders archive (Line 2), some quality measure has to be calculated (Line 3) for all the leaders to select usually one leader for each particle of the swarm. In the main loop of the algorithm, the flight of each particle is performed after a leader has been selected (Lines 6-8) and, optionally, a mutation or *turbulence* operator can be applied (Line 9); then, the particle is evaluated and its corresponding personal best is updated (Lines 10-11). After each iteration, the set of leaders is updated and the quality measure is calculated again (Lines 13-14). After the termination condition, the archive (A) is returned as the result of the search.

6.3 Studied Approaches

The studied approaches we have considered in this work can be classified as *Pareto-based* MOPSOs [RSC06]. The basic idea, commonly found in all these algorithms, is to select as leaders the particles that are non-dominated with respect to the swarm. However, this leader selection scheme can be slightly different depending on the additional information each algorithm includes on its own mechanism (e.g., information provided by a density estimator). We next summarize the main features of the considered MOPSOs:

- **Non-dominated Sorting PSO:** NSPSO [Li03] incorporates the main mechanisms of NSGA-II [DPAM02] to a PSO algorithm. In this approach, once a particle has updated its position, instead of comparing the new position only against the *pbest* position of the particle, all the *pbest* positions of the swarm and all the new positions recently obtained are combined in just one set (given a total of $2N$ solutions, where N is the size of the swarm). Then, NSPSO selects the best solutions among them to conform the next swarm (by means of a non-dominated sorting). This approach also selects the leaders randomly from the leaders set (stored in an external archive) among the best of them, based on two different mechanisms: a niche count and a nearest neighbor density estimator. This approach uses a mutation operator that is applied at each iteration step only to the particle with the smallest density estimator value.

- **SigmaMOPSO:** In SigmaMOPSO [MT03], a sigma value is assigned to each particle of the swarm and of an external archive. Then, a given particle of the swarm selects as its leader to the particle of the external archive with the closest sigma value. The use of the sigma values makes the selection pressure of PSO even higher, which may cause premature convergence in some cases. To avoid this, a turbulence operator is used, which is applied on the decision variable space.

- **Optimized MOPSO:** The main features of OMOPSO [RSC05] include the use of the crowding distance of NSGA-II to filter out leader solutions and the combination of two mutation operators to accelerate the convergence of the swarm. The original OMOPSO algorithm makes use of the concept of ϵ-dominance to limit the number of solutions produced by the algorithm. We consider here a variant discarding the use ϵ-dominance, being the leaders archive the result of the execution of the technique.

- **Another MOPSO:** AMOPSO [TC04] uses the concept of Pareto dominance to determine the flight direction of a particle. Authors adopt clustering techniques to divide the population of particles into several swarms. This aims at providing a better distribution of solutions in the decision variable space. Each sub-swarm has its own set of leaders (non-dominated particles). In each sub-swarm, a PSO algorithm is executed (leaders are randomly chosen) and, at some point, the different sub-swarms exchange information: the leaders of each swarm are migrated to a different swarm in order to variate the selection pressure. Also, this approach does not use an external archive since elitism in this case is an emergent process derived from the migration of leaders.

- **Pareto Dominance MOPSO:** in MOPSOpd [ABEF05], authors propose methods based exclusively on Pareto dominance for selecting leaders from an unconstrained non-dominated external archive. Three different selection techniques are presented: one technique that explicitly promotes diversity (called Rounds by the authors), one technique that explicitly promotes convergence (called Random), and finally one technique that is a weighted probabilistic

method (called Prob) reaching a compromise between Random and Rounds. Additionally, MOPSOpd uses a turbulence factor that is added to the position of the particles with certain probability; we have used the same operator applied in SigmaMOPSO.

- **Comprehensive Learning MOPSO:** MOCLPSO [HSL06] incorporates a Pareto dominance mechanism to the CLPSO algorithm for selecting leaders from non-dominated external archive. In this approach, a crowding distance method is used to estimate the density of the solutions just once the external archive reaches its maximum size. The distance values of all the archive members are calculated and sorted from large to small. The first $Nmax$ (maximum size of archive) members are kept whereas the remaining ones are deleted from the archive. The leaders are randomly chosen from this external archive of non-dominated solutions. In MOCLPSO, no perturbation methods are applied to keep the diversity through the evolution steps.

6.4 Experimentation

In this section, we detail the parameters settings we have used, as well as the methodology followed in the experiments. The benchmarking MOPs chosen to evaluate the six MOPSOs have been the aforementioned ZDT [ZDT00], DTLZ [DTLZ05], and WFG [HHBW06] test suites, leading to a total number of 21 problems. The two latter families of MOPs have been used with their bi-objective formulation. For assessing the performance of the algorithms, we have considered three quality indicators: additive unary epsilon indicator ($I^1_{\epsilon+}$) [KTZ06], spread (Δ) [DPAM02], and hypervolume (HV) [ZT99]. The two first indicators measure, respectively, the convergence and the diversity of the resulting Pareto fronts, while the last one measures both convergence and diversity. All the algorithms have been implemented using jMetal [DNL$^+$06], a Java-based framework for developing metaheuristics for solving multi-objective optimization problems.

6.4.1 Parameterization

We have chosen a common subset of parameter settings which are the same to all the algorithms. Thus, the size of the swarm and the leader archive, when applicable, is fixed to 100 particles, and the stopping condition is always to perform 250 iterations (yielding a total of 25,000 function evaluations). If we consider NSPSO, for example, the swarm size and the number of iterations used in [Li03] is 200 and 100, respectively. Our approach has been to establish common settings in order to make a fair comparison, keeping the rest of the parameters according to the papers where the algorithms were originally described. The parameter settings are summarized in Table 6.1 (further details are available in references).

6.4.2 Methodology

To assess the search capabilities of the algorithms, we have made 100 independent runs of each experiment, and we have obtained the median, $\tilde{x}$, and interquartile range, IQR, as measures of location (or central tendency) and statistical dispersion, respectively. Successful statistical tests are marked with '+' symbols in the last column in all the tables containing the results; conversely, '-' means that no statistical differences were found in distributions (p-value > 0.05). The best result for each problem has a gray colored background. For the sake of a better understanding of the results, we have also used a clearer grey background to indicate the second best result.

Table 6.1: Parameter settings

Common parameters	
Swarm size	100 Particles
Iterations	250
NSPSO [Li03]	
Variant	CD (Crowding distance)
C_1, C_2	2.0
w	Decreased from 1.0 to 0.4
SigmaMOPSO [MT03]	
Archive size	100
C_1, C_2	2.0
w	0.4
Mutation	$newPosition = position + rand(0.0, 1.0) * position$
Mutation probability	0.05
OMOPSO [RSC05]	
Archive size	100
C_1, C_2	$rand(1.5, 2.0)$
w	$rand(0.1, 0.5)$
Mutation	uniform + non-uniform + no mutation
Mutation probability	Each mutation is applied to 1/3 of the swarm
AMOPSO [TC04]	
Number of subswarms	5
C_1, C_2	2.0
w	0.4
MOPSOpd [ABEF05]	
Archive Size	100
C_1, C_2	1.0
w	0.5
Mutation	$newPosition = position + rand(0.0, 1.0) * position$
Mutation probability	0.05
Selection method	Rounds
MOCLPSO [HSL06]	
Archive Size	100
C_1, C_2	N/A
w	0.9 to 0.2

Table 6.2: Median and interquartile range of the $I^1_{\epsilon+}$ quality indicator

Problem	NSPSO $\tilde{x}_{IQR}$	SigmaMOPSO $\tilde{x}_{IQR}$	OMOPSO $\tilde{x}_{IQR}$	AMOPSO $\tilde{x}_{IQR}$	MOPSOpd $\tilde{x}_{IQR}$	MOCLPSO $\tilde{x}_{IQR}$	
ZDT1	$4.57e-1_{3.7e-1}$	$3.07e-2_{2.6e-2}$	$6.36e-3_{5.1e-4}$	$2.41e-1_{8.0e-2}$	$6.75e-2_{1.6e-2}$	$3.74e-1_{8.8e-2}$	+
ZDT2	$1.54e+0_{8.5e-1}$	$1.00e+0_{0.0e+0}$	$6.19e-3_{5.4e-4}$	$6.33e-1_{8.3e-1}$	$1.00e+0_{8.9e-1}$	$6.45e-1_{1.4e-1}$	+
ZDT3	$9.14e-1_{4.1e-1}$	$9.75e-1_{8.3e-1}$	$1.32e-2_{7.7e-3}$	$7.30e-1_{3.5e-1}$	$1.66e-1_{1.1e-1}$	$5.97e-1_{2.0e-1}$	+
ZDT4	$4.14e+1_{1.6e+1}$	$8.30e+0_{6.8e+0}$	$5.79e+0_{4.3e+0}$	$1.21e+1_{7.6e+0}$	$4.23e+0_{2.1e+0}$	$1.71e+1_{1.3e+1}$	+
ZDT6	$1.81e-1_{3.2e-1}$	$5.91e-3_{1.1e-3}$	$4.65e-3_{4.2e-4}$	$1.69e-1_{6.0e-2}$	$1.21e-1_{7.0e-2}$	$3.38e+0_{3.8e-1}$	+
DTLZ1	$2.30e+1_{8.0e+0}$	$2.54e+1_{1.3e+1}$	$1.92e+1_{1.1e+1}$	$8.46e+0_{1.9e+1}$	$1.72e+1_{1.1e+1}$	$2.12e+1_{8.0e+0}$	+
DTLZ2	$4.41e-2_{6.5e-2}$	$1.13e-1_{9.1e-2}$	$6.72e-3_{9.1e-4}$	$1.25e-1_{3.9e-2}$	$9.26e-2_{5.1e-2}$	$3.95e-2_{3.8e-2}$	+
DTLZ3	$1.04e+2_{6.2e+1}$	$1.79e+2_{7.5e+1}$	$8.86e+1_{9.5e+1}$	$4.41e+1_{9.0e+1}$	$1.23e+2_{6.5e+1}$	$2.37e+2_{5.7e+1}$	+
DTLZ4	$8.91e-2_{5.9e-2}$	$3.00e-1_{4.5e-2}$	$3.18e-2_{1.0e-2}$	$2.20e-1_{1.1e-1}$	$6.33e-2_{3.0e-2}$	$2.56e-2_{8.6e-3}$	+
DTLZ5	$3.92e-2_{3.6e-2}$	$1.11e-1_{9.8e-2}$	$6.62e-3_{8.9e-4}$	$1.22e-1_{4.3e-2}$	$9.10e-2_{4.0e-2}$	$3.31e-2_{3.0e-2}$	+
DTLZ6	$1.47e+0_{7.9e-1}$	$1.00e+0_{2.9e-1}$	$5.36e-3_{4.8e-4}$	$1.75e-1_{9.1e-1}$	$1.57e+0_{1.3e+0}$	$4.77e+0_{3.2e-1}$	+
DTLZ7	$1.33e+0_{1.4e+0}$	$1.27e+0_{2.7e-2}$	$7.13e-3_{6.8e-4}$	$3.00e-1_{1.9e-1}$	$1.65e-1_{1.1e-1}$	$4.94e-1_{1.0e-1}$	+
WFG1	$1.36e+0_{7.7e-2}$	$1.00e+0_{9.3e-2}$	$1.35e+0_{4.9e-2}$	$1.53e+0_{3.0e-2}$	$1.10e+0_{2.0e-1}$	$1.31e+0_{5.1e-2}$	+
WFG2	$1.67e-2_{5.5e-3}$	$4.87e-2_{3.6e-2}$	$1.04e-2_{1.7e-3}$	$3.57e-1_{1.8e-1}$	$7.24e-2_{2.1e-2}$	$5.96e-2_{3.7e-2}$	+
WFG3	$2.00e+0_{5.3e-4}$	$2.00e+0_{4.2e-3}$	$2.00e+0_{1.6e-5}$	$2.10e+0_{1.2e-1}$	$2.00e+0_{4.5e-5}$	$2.12e+0_{2.0e-1}$	+
WFG4	$1.09e-1_{1.8e-2}$	$6.06e-2_{2.7e-2}$	$5.98e-2_{1.5e-2}$	$3.21e-1_{8.1e-2}$	$5.57e-2_{1.8e-2}$	$8.04e-2_{2.4e-2}$	+
WFG5	$8.34e-2_{2.0e-2}$	$6.36e-2_{1.2e-3}$	$6.37e-2_{9.0e-4}$	$6.24e-1_{3.3e-1}$	$3.24e-1_{3.5e-1}$	$2.57e-1_{2.2e-1}$	+
WFG6	$1.04e-1_{6.6e-2}$	$5.60e-1_{3.8e-1}$	$1.79e-2_{2.5e-3}$	$4.63e-1_{1.3e-1}$	$3.30e-1_{2.6e-1}$	$2.40e-1_{2.3e-1}$	+
WFG7	$4.05e+2_{6.1e+3}$	$5.75e+2_{1.8e+2}$	$1.94e+2_{1.7e+3}$	$3.77e+1_{1.5e+1}$	$6.16e+1_{1.1e+1}$	$2.44e+1_{3.4e+1}$	+
WFG8	$5.24e-1_{9.2e-2}$	$5.66e-1_{1.9e-1}$	$5.06e-1_{3.4e-2}$	$8.30e-1_{1.2e-1}$	$5.39e-1_{2.3e-2}$	$7.70e-1_{6.0e-2}$	+
WFG9	$6.38e-2_{2.0e-2}$	$2.89e-2_{1.7e-3}$	$2.95e-2_{2.5e-3}$	$3.25e-1_{2.5e-1}$	$1.11e-1_{4.6e-2}$	$1.49e-1_{2.1e-1}$	+

6.5 Computational Results

This section is devoted to evaluating and analyzing the results of the experiments. We start by analyzing the values obtained after applying the $I^1_{\epsilon+}$ quality indicator, which are contained in Table 6.2. We can observe that OMOPSO clearly outperforms the rest of MOPSOs according to this indicator, achieving the lowest (best) values in 13 out of the 21 problems composing the

benchmark. It also obtains six second best values. The next best performing algorithms are SigmaMOPSO, MOPSOpd, and AMOPSO, which get similar numbers of best and second best results. Thus, we can claim that OMOPSO produces solution sets having better convergence to the Pareto fronts in most of the benchmark problems considered in our study. All the results have statistical significance, as it can be seen in the last column, where only '+ ' symbols are seen.

The values obtained after applying the Δ quality indicator are included in Table 6.3. We can observe again that OMOPSO is clearly the best performing algorithm, yielding the lowest (best) values in 16 out of the 21 problems. Considering the next algorithms according to the best and second best indicator values, we find SigmaMOPSO, NSPSO, and MOCLPSO. AMOPSO is the worst performer according to the Δ indicator, not achieving any best nor second best result.

Table 6.3: Median and interquartile range of the Δ quality indicator

Problem	NSPSO $\tilde{x}_{IQR}$	SigmaMOPSO $\tilde{x}_{IQR}$	OMOPSO $\tilde{x}_{IQR}$	AMOPSO $\tilde{x}_{IQR}$	MOPSOpd $\tilde{x}_{IQR}$	MOCLPSO $\tilde{x}_{IQR}$	
ZDT1	$7.19e-1_{1.0e-1}$	$4.11e-1_{3.9e-1}$	$1.00e-1_{1.4e-2}$	$9.57e-1_{1.6e-1}$	$6.03e-1_{1.1e-1}$	$7.70e-1_{6.4e-2}$	+
ZDT2	$9.82e-1_{9.4e-2}$	$1.00e+0_{0.0e+0}$	$9.45e-2_{1.8e-2}$	$1.00e+0_{6.0e-2}$	$1.00e+0_{2.8e-1}$	$8.03e-1_{7.4e-2}$	+
ZDT3	$8.17e-1_{9.7e-2}$	$1.09e+0_{3.6e-1}$	$7.35e-1_{5.2e-2}$	$9.00e-1_{1.5e-1}$	$8.59e-1_{6.7e-2}$	$8.85e-1_{5.7e-2}$	+
ZDT4	$9.53e-1_{8.0e-2}$	$1.00e+0_{3.3e-3}$	$8.78e-1_{5.2e-2}$	$1.03e+0_{2.5e-2}$	$1.00e+0_{2.4e-2}$	$9.32e-1_{8.2e-2}$	+
ZDT6	$1.39e+0_{6.6e-2}$	$2.89e-1_{3.6e-1}$	$8.78e-2_{1.2e+0}$	$1.12e+0_{1.5e-1}$	$1.20e+0_{2.7e-1}$	$9.67e-1_{4.1e-2}$	+
DTLZ1	$8.38e-1_{1.2e-1}$	$1.14e+0_{1.7e-1}$	$7.77e-1_{1.1e-1}$	$1.13e+0_{2.6e-1}$	$8.72e-1_{2.0e-1}$	$7.90e-1_{7.2e-2}$	+
DTLZ2	$6.02e-1_{1.5e-1}$	$1.01e+0_{1.4e-1}$	$1.81e-1_{2.3e-2}$	$1.15e+0_{1.8e-1}$	$1.21e+0_{8.6e-2}$	$7.92e-1_{8.7e-2}$	+
DTLZ3	$9.31e-1_{2.0e-1}$	$1.23e+0_{1.6e-1}$	$7.90e-1_{1.1e-1}$	$1.09e+0_{4.3e-1}$	$8.55e-1_{1.3e-1}$	$7.69e-1_{8.5e-2}$	+
DTLZ4	$7.17e-1_{1.7e-1}$	$1.41e+0_{8.0e-1}$	$6.77e-1_{7.9e-2}$	$1.46e+0_{2.7e-1}$	$1.10e+0_{9.2e-2}$	$7.33e-1_{5.3e-2}$	+
DTLZ5	$5.99e-1_{9.3e-2}$	$1.00e+0_{1.7e-1}$	$1.77e-1_{2.6e-2}$	$1.16e+0_{1.9e-1}$	$1.21e+0_{9.3e-2}$	$7.89e-1_{8.9e-2}$	+
DTLZ6	$8.18e-1_{4.0e-1}$	$1.28e+0_{1.0e+0}$	$1.18e-1_{1.7e-2}$	$1.23e+0_{4.4e-1}$	$8.35e-1_{1.5e-1}$	$8.04e-1_{7.2e-2}$	+
DTLZ7	$9.08e-1_{1.6e-1}$	$7.96e-1_{2.4e-1}$	$5.21e-1_{6.8e-3}$	$1.02e+0_{2.4e-1}$	$7.95e-1_{1.3e-1}$	$8.51e-1_{7.0e-2}$	+
WFG1	$1.14e+0_{5.5e-2}$	$7.50e-1_{1.2e-1}$	$1.17e+0_{6.0e-2}$	$1.30e+0_{3.9e-2}$	$1.16e+0_{7.8e-2}$	$1.12e+0_{4.2e-2}$	+
WFG2	$8.65e-1_{9.0e-2}$	$9.61e-1_{8.5e-2}$	$7.64e-1_{5.5e-3}$	$9.94e-1_{1.9e-1}$	$1.22e+0_{7.0e-2}$	$1.11e+0_{5.8e-2}$	+
WFG3	$5.00e-1_{2.6e-2}$	$4.96e-1_{2.5e-2}$	$3.78e-1_{8.7e-3}$	$1.20e+0_{8.7e-2}$	$1.19e+0_{1.3e-1}$	$9.04e-1_{6.2e-2}$	+
WFG4	$6.25e-1_{5.0e-2}$	$5.01e-1_{7.7e-2}$	$5.06e-1_{6.3e-2}$	$1.14e+0_{1.3e-1}$	$4.83e-1_{4.4e-2}$	$6.18e-1_{4.9e-2}$	+
WFG5	$3.59e-1_{4.5e-2}$	$1.44e-1_{2.0e-2}$	$1.44e-1_{2.0e-2}$	$1.03e+0_{1.7e-1}$	$1.13e+0_{2.3e-1}$	$8.06e-1_{9.7e-2}$	+
WFG6	$5.98e-1_{8.1e-2}$	$6.34e-1_{2.1e-1}$	$1.63e-1_{2.5e-2}$	$1.09e+0_{1.7e-1}$	$1.23e+0_{7.0e-2}$	$8.32e-1_{7.6e-2}$	+
WFG7	$3.71e+1_{5.8e+2}$	$4.07e+1_{5.5e+2}$	$1.59e+1_{2.1e+2}$	$1.13e+0_{1.3e+1}$	$1.31e+0_{7.1e+2}$	$9.13e+1_{8.7e+2}$	+
WFG8	$7.19e-1_{8.4e-2}$	$9.08e-1_{1.7e-1}$	$7.93e-1_{8.8e-2}$	$1.02e+0_{1.4e-1}$	$8.68e-1_{6.6e-2}$	$7.88e-1_{5.3e-2}$	+
WFG9	$5.07e-1_{1.3e-1}$	$2.22e-1_{2.6e-2}$	$2.24e-1_{2.7e-2}$	$1.19e+0_{1.5e-1}$	$7.54e-1_{5.2e-2}$	$7.29e-1_{6.3e-2}$	+

After applying a quality indicator that measures convergence and another one that measures diversity, the HV indicator should confirm the previous results. The HV values, included in Table 6.4, show that OMOPSO generates solution sets with the highest (best) values in 15 out of the 21 problems. Thus, we can state that according to the parameterization, quality indicators, and benchmark problems considered in this work, OMOPSO is clearly the most salient technique among the six considered in our study.

The results corresponding to problems ZDT4, DTLZ1, and DTLZ3 deserve additional comments. We have used the '–' symbol in Table 6.4 to indicate those experiments in which the HV value is equal to 0, meaning that the solution sets obtained by the algorithms are outside the limits of the Pareto front; when applying the HV indicator these solutions are not taken into account, because otherwise the obtained results would be unreliable. In the case of the three aforementioned problems, none of the six algorithms is able to achieve a HV greater than 0 over the 100 independent runs. We can also see that other problems are difficult to solve by some techniques, e.g., ZDT2 and DTLZ6. The statistical tests indicate that the results of the Δ and HV indicators have statistical confidence. To provide further statistical information, we show in Table 6.5 those problems for which no statistical differences appear between OMOPSO and the rest of algorithms considering the three quality indicators. It can be observed that statistical differences exist for most of the pair-wise comparisons.

Table 6.4: Median and interquartile range of the HV quality indicator

Problem	NSPSO $\bar{x}_{IQR}$	SigmaMOPSO $\bar{x}_{IQR}$	OMOPSO $\bar{x}_{IQR}$	AMOPSO $\bar{x}_{IQR}$	MOPSOpd $\bar{x}_{IQR}$	MOCLPSO $\bar{x}_{IQR}$	
ZDT1	$1.54e-1_{2.4e-1}$	$6.54e-1_{8.3e-3}$	$6.61e-1_{1.5e-4}$	$3.81e-1_{9.3e-2}$	$5.94e-1_{1.7e-2}$	$3.28e-1_{4.6e-2}$	+
ZDT2	–	–	$3.28e-1_{2.5e-4}$	$4.10e-2_{1.9e-1}$	$0.00e+0_{2.6e-1}$	$6.54e-2_{3.7e-2}$	+
ZDT3	$1.12e-1_{1.2e-1}$	$3.21e-1_{2.3e-1}$	$5.10e-1_{3.8e-3}$	$2.45e-1_{1.1e-1}$	$4.38e-1_{7.2e-2}$	$2.55e-1_{3.2e-2}$	+
ZDT4	–	–	–	–	–	–	-
ZDT6	$3.09e-1_{1.3e-1}$	$4.01e-1_{3.1e-4}$	$4.01e-1_{1.5e-4}$	$2.31e-1_{4.1e-2}$	$3.50e-1_{5.7e-2}$	–	+
DTLZ1	–	–	–	–	–	–	-
DTLZ2	$1.64e-1_{5.9e-2}$	$1.64e-1_{2.1e-2}$	$2.10e-1_{4.5e-4}$	$1.23e-1_{2.4e-2}$	$1.78e-1_{2.5e-2}$	$2.01e-1_{2.3e-3}$	+
DTLZ3	–	–	–	–	–	–	-
DTLZ4	$1.37e-1_{5.1e-2}$	–	$1.96e-1_{6.1e-3}$	$7.62e-2_{9.8e-2}$	$1.90e-1_{9.8e-3}$	$1.96e-1_{4.0e-3}$	+
DTLZ5	$1.71e-1_{3.5e-2}$	$1.65e-1_{2.3e-2}$	$2.11e-1_{5.4e-4}$	$1.22e-1_{2.9e-2}$	$1.77e-1_{2.0e-2}$	$2.01e-1_{2.1e-3}$	+
DTLZ6	–	–	$2.12e-1_{4.4e-5}$	$8.77e-2_{1.5e-1}$	–	–	+
DTLZ7	$1.59e-2_{9.7e-2}$	$2.18e-1_{1.7e-2}$	$3.34e-1_{3.2e-4}$	$2.00e-1_{7.1e-2}$	$2.53e-1_{5.5e-2}$	$1.01e-1_{1.3e-2}$	+
WFG1	$8.98e-2_{8.3e-3}$	$1.21e-1_{2.2e-3}$	$1.04e-1_{1.0e-2}$	$6.22e-2_{7.4e-3}$	$1.69e-1_{7.2e-2}$	$1.01e-1_{5.1e-3}$	+
WFG2	$5.61e-1_{2.5e-3}$	$5.60e-1_{1.7e-3}$	$5.64e-1_{1.0e-4}$	$4.68e-1_{3.9e-2}$	$5.57e-1_{3.6e-3}$	$5.60e-1_{1.8e-3}$	+
WFG3	$4.40e-1_{3.3e-4}$	$4.38e-1_{8.0e-4}$	$4.42e-1_{5.4e-5}$	$4.04e-1_{1.2e-2}$	$4.27e-1_{1.8e-2}$	$4.30e-1_{1.3e-2}$	+
WFG4	$1.78e-1_{7.0e-3}$	$2.00e-1_{1.6e-3}$	$2.02e-1_{1.6e-3}$	$1.27e-1_{1.2e-2}$	$2.07e-1_{1.3e-3}$	$2.00e-1_{2.3e-3}$	+
WFG5	$1.96e-1_{2.8e-4}$	$1.96e-1_{8.8e-5}$	$1.96e-1_{6.3e-5}$	$1.60e-1_{1.7e-2}$	$1.68e-1_{5.9e-2}$	$1.90e-1_{1.9e-3}$	+
WFG6	$1.75e-1_{2.6e-2}$	$1.90e-1_{1.9e-2}$	$2.09e-1_{3.5e-4}$	$9.88e-2_{2.8e-2}$	$1.60e-1_{4.7e-2}$	$2.01e-1_{1.9e-3}$	+
WFG7	$2.03e+1_{2.7e+3}$	$2.02e+1_{1.1e+3}$	$2.09e+1_{1.7e+4}$	$1.14e+1_{1.4e+2}$	$9.49e+2_{4.2e+2}$	$2.01e+1_{2.7e+3}$	+
WFG8	$1.07e-1_{8.7e-3}$	$1.33e-1_{4.2e-3}$	$1.26e-1_{3.0e-3}$	$6.08e-2_{1.9e-2}$	$1.41e-1_{3.0e-3}$	$1.33e-1_{1.9e-3}$	+
WFG9	$2.24e-1_{6.1e-3}$	$2.34e-1_{4.1e-4}$	$2.34e-1_{6.6e-4}$	$1.87e-1_{1.1e-2}$	$2.29e-1_{4.7e-3}$	$2.30e-1_{1.1e-3}$	+

Table 6.5: Benchmark problems for which no statistical differences were found between OMOPSO and the rest of the algorithms

	$I^1_{\epsilon+}$	Δ	HV
AMOPSO	-	-	-
	DTLZ3	-	-
	-	-	-
MOCLPSO	-	ZDT6	-
	DTLZ1, DTLZ4	DTLZ1, DTLZ3	DTLZ4
	-	WFG8	WFG1, WFG4
MOPSOpd	ZDT4	-	-
	DTLZ1, DTLZ3	-	-
	WFG3, WFG4	WFG1, WFG4	-
NSPSO	-	-	-
	DTLZ3	DTLZ4	-
	WFG1, WFG8	-	-
SigmaMOPSO	-	ZDT6	-
	-	-	-
	WFG4, WFG5, WFG9	WFG4, WFG5, WFG9	WFG5, WFG9

6.6 Discussion

The conclusion drawn from the analysis of the results in the previous section is that OMOPSO performs the best in our study. In this section, we carry out the same experiments but using OMOPSO and NSGA-II in order to put the results of the first one in context. Such a comparison will allow us to know how competitive OMOPSO is. Before that, we investigate why OMOPSO (as well as the rest of MOPSOs) is unable to solve the ZDT4, DTLZ1, and DTLZ3 problems. If we consider ZDT4, it is a well-known problem characterized by having many local optima (it is a multifrontal problem). We have traced the velocity of the second variable in the first particle in OMOPSO when facing the solution of ZDT4 (the second variable takes values in the interval $[-5, +5]$, which provides a better illustration of the following analysis than using the first variable, which ranges in $[0, 1]$). The obtained values after the 250 iterations are depicted in Figure 6.1. We can observe that the velocity (speed) values suffer a kind of erratic behavior in some points of the execution, alternating very high with very low values. Let us note that the limits of the second variable in ZDT4 are $[-5, +5]$, and the velocity takes values higher than ± 20. The consequence is that this particle is moving to its extreme values continuously, so it is not contributing to guide the search.

Figure 6.1: Tracing the velocity of the second variable of OMOPSO when solving ZDT4

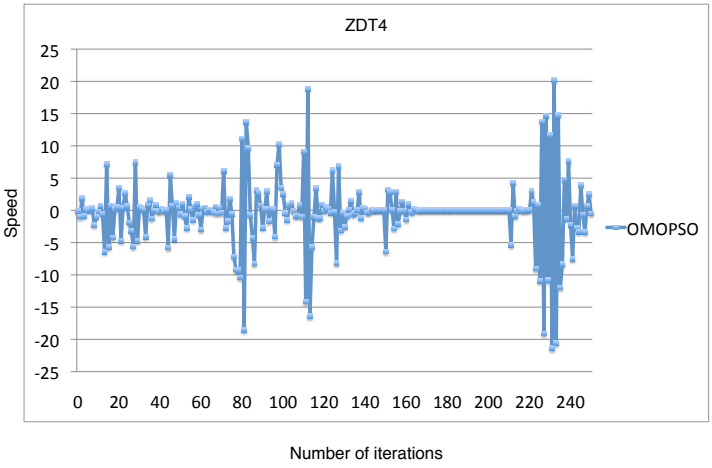
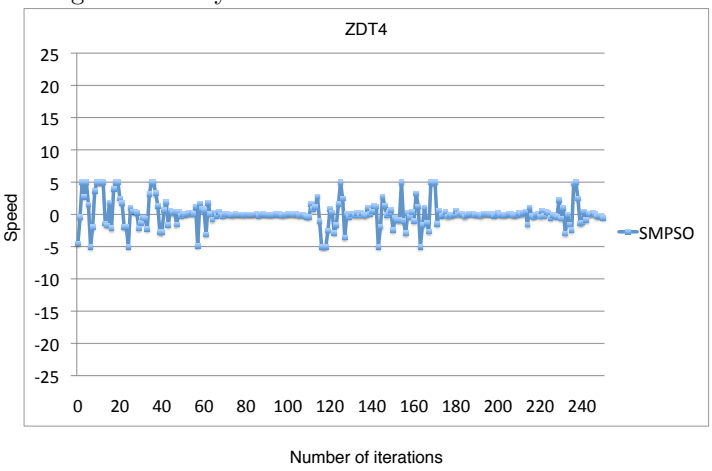


Figure 6.2: Tracing the velocity of the second variable of SMPSO when solving ZDT4



To find out whether this is one of the reasons making OMOPSO unable to solve multi frontal MOPs, we have modified it by including a velocity constraint mechanism, similar to the one proposed in [CK02]. In addition, the accumulated velocity of each variable j (in each particle) is also bounded by means of the following equation:

$$v_{i,j}(t) = \begin{cases} delta_j & \text{if } v_{i,j}(t) > delta_j \\ -delta_j & \text{if } v_{i,j}(t) \leq -delta_j \\ v_{i,j}(t) & \text{otherwise} \end{cases} \tag{6.1}$$

where

$$delta_j = \frac{(upper\_limit_j - lower\_limit_j)}{2} \tag{6.2}$$

This way, we can ensure an effective new position calculation. We have called the resulting algorithm SMPSO (Speed-constrained Multi-objective PSO). In Figure 6.2 we show again the velocity of the particle representing the second parameter of ZDT4. We can observe that the erratic movements of the velocity have vanished, so the particle is taking values inside the bounds of the variable and thus it is moving along different regions of the search space. To evaluate the effect of the changes in SMPSO, we have included this algorithm in the comparison between OMOPSO and NSGA-II. We have solved all the problems again, following the same methodology. The parameter settings of NSGA-II are: the population size is 100 individuals, we have used SBX and polynomial mutation [Deb01] as operators for crossover and mutation operators, respectively, and the distribution indexes for both operators are $\eta_c = 20$ and $\eta_m = 20$, respectively. The crossover probability is $p_c = 0.9$ and the mutation probability is $p_m = 1/L$, where L is the number of decision variables.

Table 6.6: NSGA-II vs OMOPSO vs SMPSO: Median and interquartile range of $I^1_{\epsilon+}$

Problem	NSGA-II $\bar{x}_{IQR}$	OMOPSO $\bar{x}_{IQR}$	SMPSO $\bar{x}_{IQR}$	
ZDT1	$1.37e-2_{3.0e-3}$	$6.36e-3_{5.1e-4}$	$5.78e-3_{3.8e-4}$	+
ZDT2	$1.28e-2_{2.3e-3}$	$6.19e-3_{5.4e-4}$	$5.66e-3_{3.0e-4}$	+
ZDT3	$8.13e-3_{1.9e-3}$	$1.32e-2_{7.7e-3}$	$6.09e-3_{1.3e-3}$	+
ZDT4	$1.49e-2_{3.0e-3}$	$5.79e+0_{4.3e+0}$	$7.93e-3_{1.4e-3}$	+
ZDT6	$1.47e-2_{2.8e-3}$	$4.65e-3_{4.2e-4}$	$4.87e-3_{4.8e-4}$	+
DTLZ1	$7.13e-3_{1.6e-3}$	$1.92e+1_{1.1e+1}$	$3.73e-3_{5.4e-4}$	+
DTLZ2	$1.11e-2_{2.7e-3}$	$6.72e-3_{9.1e-4}$	$5.81e-3_{6.0e-4}$	+
DTLZ3	$1.04e+0_{1.2e+0}$	$8.86e+1_{9.5e+1}$	$6.57e-3_{1.0e-2}$	+
DTLZ4	$1.13e-2_{9.9e-1}$	$3.18e-2_{1.0e-2}$	$6.54e-3_{8.8e-4}$	+
DTLZ5	$1.05e-2_{2.5e-3}$	$6.62e-3_{8.9e-4}$	$5.77e-3_{6.1e-4}$	+
DTLZ6	$4.39e-2_{3.4e-2}$	$5.36e-3_{4.8e-4}$	$5.22e-3_{4.4e-4}$	+
DTLZ7	$1.04e-2_{2.8e-3}$	$7.13e-3_{6.8e-4}$	$5.46e-3_{4.3e-4}$	+
WFG1	$3.52e-1_{4.6e-1}$	$1.35e+0_{4.9e-2}$	$1.34e+0_{4.6e-2}$	+
WFG2	$7.10e-1_{7.0e-1}$	$1.04e-2_{1.7e-3}$	$1.40e-2_{3.4e-3}$	+
WFG3	$2.00e+0_{5.8e-4}$	$2.00e+0_{1.6e-5}$	$2.00e+0_{3.9e-4}$	+
WFG4	$3.26e-2_{6.7e-3}$	$5.98e-2_{1.5e-2}$	$6.46e-2_{6.0e-3}$	+
WFG5	$8.41e-2_{8.3e-3}$	$6.37e-2_{9.0e-4}$	$6.40e-2_{2.0e-3}$	+
WFG6	$4.14e-2_{1.6e-2}$	$1.79e-2_{2.5e-3}$	$2.56e-2_{3.8e-3}$	+
WFG7	$3.47e+2_{8.1e+3}$	$1.94e+2_{1.7e+3}$	$2.67e+2_{3.8e+3}$	+
WFG8	$3.38e-1_{2.3e-1}$	$5.06e-1_{3.4e-2}$	$4.32e-1_{7.8e-2}$	+
WFG9	$3.73e-2_{7.5e-3}$	$2.95e-2_{2.5e-3}$	$3.15e-2_{3.3e-3}$	+

In Table 6.6, we include the median and interquartile range of NSGA-II, OMOPSO, and SMPSO corresponding to the $I^1_{\epsilon+}$ quality indicator. We observe that SMPSO yields the best values in 11 out of the 12 problems comprising the ZDT and DTLZ benchmarks. If we focus on the WFG problems, the lowest (best) metric values are shared between OMOPSO (six problems) and NSGA-II (three problems), while SMPSO obtains the second lowest values in 8 out of the 9 WFG problems. These results indicate first, that OMOPSO is competitive when compared against NSGA-II concerning convergence and, second, that the velocity constraint mechanism included in SMPSO improves globally the behavior of OMOPSO considering all the benchmark problems.

The values obtained when applying the Δ and HV indicators are included in Tables 6.7 and 6.8, respectively. We can observe that we can practically draw the same conclusions obtained from the $I^1_{\epsilon+}$ indicator, i.e., the algorithms obtain the lowest values in the same problems according to the convergence and diversity indicators. In all the experiments included in this section all the statistical tests are significant, which actually grounds our claims. If we focus in the HV and in those problems in which OMOPSO obtained a value of 0 (ZDT4, DTLZ1, and DTLZ3), we see that the velocity constraint mechanism added to SMPSO allows it to successfully solve them. NSGA-II also outperforms OMOPSO in this sense, only presenting difficulties in DTLZ3.

Table 6.7: NSGA-II vs OMOPSO vs SMPSO: Median and interquartile range of Δ

Problem	NSGA-II $\bar{x}_{IQR}$	OMOPSO $\bar{x}_{IQR}$	SMPSO $\bar{x}_{IQR}$	
ZDT1	$3.70e-1_{4.2e-2}$	$1.00e-1_{1.4e-2}$	$8.66e-2_{1.6e-2}$	+
ZDT2	$3.81e-1_{4.7e-2}$	$9.45e-2_{1.8e-2}$	$7.46e-2_{1.5e-2}$	+
ZDT3	$7.47e-1_{1.8e-2}$	$7.35e-1_{5.2e-2}$	$7.17e-1_{1.7e-2}$	+
ZDT4	$4.02e-1_{5.8e-2}$	$8.78e-1_{5.2e-2}$	$1.53e-1_{2.2e-2}$	+
ZDT6	$3.56e-1_{3.6e-2}$	$8.78e-2_{1.2e+0}$	$7.28e-1_{1.2e+0}$	+
DTLZ1	$4.03e-1_{6.1e-2}$	$7.77e-1_{1.1e-1}$	$1.14e-1_{1.8e-2}$	+
DTLZ2	$3.84e-1_{3.8e-2}$	$1.81e-1_{2.3e-2}$	$1.59e-1_{2.3e-2}$	+
DTLZ3	$9.53e-1_{1.6e-1}$	$7.90e-1_{1.1e-1}$	$1.98e-1_{3.3e-1}$	+
DTLZ4	$3.95e-1_{6.4e-1}$	$6.77e-1_{7.9e-2}$	$1.70e-1_{2.5e-2}$	+
DTLZ5	$3.79e-1_{4.0e-2}$	$1.77e-1_{2.6e-2}$	$1.58e-1_{2.2e-2}$	+
DTLZ6	$8.64e-1_{3.0e-1}$	$1.18e-1_{1.7e-2}$	$1.14e-1_{2.1e-2}$	+
DTLZ7	$6.23e-1_{2.5e-2}$	$5.21e-1_{6.8e-3}$	$5.20e-1_{2.0e-3}$	+
WFG1	$7.18e-1_{5.4e-2}$	$1.17e+0_{6.0e-2}$	$1.12e+0_{5.0e-2}$	+
WFG2	$7.93e-1_{1.7e-2}$	$7.64e-1_{5.5e-3}$	$8.26e-1_{3.5e-2}$	+
WFG3	$6.12e-1_{3.6e-2}$	$3.78e-1_{8.7e-3}$	$3.84e-1_{6.4e-3}$	+
WFG4	$3.79e-1_{3.9e-2}$	$5.06e-1_{6.3e-2}$	$5.51e-1_{7.0e-2}$	+
WFG5	$4.13e-1_{5.1e-2}$	$1.44e-1_{2.0e-2}$	$1.50e-1_{2.8e-2}$	+
WFG7	$3.79e+1_{4.6e+2}$	$1.59e+1_{2.1e+2}$	$2.44e+1_{3.1e+2}$	+
WFG6	$3.90e-1_{4.2e-2}$	$1.63e-1_{2.5e-2}$	$2.47e-1_{4.1e-2}$	+
WFG8	$6.45e-1_{5.5e-2}$	$7.93e-1_{8.8e-2}$	$8.08e-1_{5.4e-2}$	+
WFG9	$3.96e-1_{4.1e-2}$	$2.24e-1_{2.7e-2}$	$2.46e-1_{2.8e-2}$	+

Table 6.8: NSGA-II vs OMOPSO vs SMPSO: Median and interquartile range of HV

Problem	NSGA-II $\bar{x}_{IQR}$	OMOPSO $\bar{x}_{IQR}$	SMPSO $\bar{x}_{IQR}$	
ZDT1	$6.59e-1_{4.4e-4}$	$6.61e-1_{1.5e-4}$	$6.62e-1_{1.5e-4}$	+
ZDT2	$3.26e-1_{4.3e-4}$	$3.28e-1_{2.5e-4}$	$3.28e-1_{1.1e-4}$	+
ZDT3	$5.15e-1_{2.3e-4}$	$5.10e-1_{3.8e-3}$	$5.15e-1_{5.1e-4}$	+
ZDT4	$6.56e-1_{4.5e-3}$	–	$6.61e-1_{3.8e-4}$	+
ZDT6	$3.88e-1_{2.3e-3}$	$4.01e-1_{1.5e-4}$	$4.01e-1_{1.0e-4}$	+
DTLZ1	$4.88e-1_{5.5e-3}$	–	$4.94e-1_{3.4e-4}$	+
DTLZ2	$2.11e-1_{3.1e-4}$	$2.10e-1_{4.5e-4}$	$2.12e-1_{2.3e-4}$	+
DTLZ3	–	–	$2.12e-1_{2.8e-3}$	+
DTLZ4	$2.09e-1_{2.1e-1}$	$1.96e-1_{6.1e-3}$	$2.09e-1_{3.3e-4}$	+
DTLZ5	$2.11e-1_{3.5e-4}$	$2.11e-1_{5.4e-4}$	$2.12e-1_{2.1e-4}$	+
DTLZ6	$1.75e-1_{3.6e-2}$	$2.12e-1_{4.4e-5}$	$2.12e-1_{4.8e-5}$	+
DTLZ7	$3.33e-1_{2.1e-4}$	$3.34e-1_{3.2e-4}$	$3.34e-1_{7.3e-5}$	+
WFG1	$5.23e-1_{1.3e-1}$	$1.04e-1_{1.0e-2}$	$9.70e-2_{5.3e-3}$	+
WFG2	$5.61e-1_{2.8e-3}$	$5.64e-1_{1.0e-4}$	$5.62e-1_{5.7e-4}$	+
WFG3	$4.41e-1_{3.2e-4}$	$4.42e-1_{5.4e-5}$	$4.41e-1_{1.1e-4}$	+
WFG4	$2.17e-1_{4.9e-4}$	$2.02e-1_{1.6e-3}$	$1.96e-1_{2.0e-3}$	+
WFG5	$1.95e-1_{3.6e-4}$	$1.96e-1_{6.3e-5}$	$1.96e-1_{5.8e-5}$	+
WFG6	$2.03e-1_{9.0e-3}$	$2.09e-1_{3.5e-4}$	$2.05e-1_{1.1e-3}$	+
WFG7	$2.09e+1_{3.3e+4}$	$2.09e+1_{1.7e+4}$	$2.06e+1_{8.2e+4}$	+
WFG8	$1.47e-1_{2.1e-3}$	$1.26e-1_{3.0e-3}$	$1.40e-1_{1.9e-3}$	+
WFG9	$2.37e-1_{1.7e-3}$	$2.34e-1_{6.6e-4}$	$2.33e-1_{4.1e-4}$	+

Table 6.9 contains those problems from which no statistical differences exist considering the three algorithms and the three quality indicators. The results of OMOPSO against NSGA-II are significant in all the problems but DTLZ3 with respect to the Δ indicator. Concerning SMPSO, there a few cases where the results are not statistically different, but they do not alter the conclusions drawn before.

We can summarize this section by stating that OMOPSO, the most salient of the six MOPSOs studied in this work, is a competitive algorithm when compared with NSGA-II, and we have shown that its search capabilities can be improved by including a velocity constraint mechanism. However, although SMPSO outperforms both NSGA-II and OMOPSO in the ZDT and DTLZ problems, it does not achieve the best result in the WFG benchmark. This indicates that more research has

Table 6.9: Behcnmark problems for which no statistical differences were found among NSGA-II, OMOPSO, and SMPSO

Quality Indicator	Algorithm	OMOPSO	SMPSO
$I^1_{\epsilon+}$	NSGA-II	N/A	WFG3, WFG8
	OMOPSO		ZDT6, DTLZ6, WFG1, WFG4
Δ	NSGA-II	DTLZ3	WFG2
	OMOPSO	N/A	ZDT6, DTLZ6
HV	NSGA-II	N/A	ZDT6
	OMOPSO		DTLZ6, DTLZ7, WFG8

to be done. It is also necessary to consider a broader set of problems as well as studying in more depth the effect of modulating the speed in a MOPSO.

6.7 Conclusions

We have evaluated six MOPSO algorithms over a set of three well-known benchmark problems by using three different quality indicators. For each experiment, 100 independent runs have been carried out, and statistical tests have been applied to know more about the confidence of the obtained results. In the context of the problems analyzed, the experimentation methodology, and the parameter settings used, we can state that OMOPSO is clearly the most salient of the six compared algorithms. The results have also shown that all the algorithms are unable to find accurate Pareto fronts for three multi frontal problems. We have studied this issue and we have proposed the use of a velocity constraint mechanism to enhance the search capability in order to solve these problems. The resulting algorithm, SMPSO, shows significant improvements when compared with respect to OMOPSO and NSGA-II. As part of our future work, we plan to study the convergence speed of MOPSO algorithms in order to determine whether they are faster than other multi-objective evolutionary algorithms in reaching the Pareto front of a problem.

Chapter 7

PSO6: Quasi-optimally Informed PSO

7.1 Introduction

In this chapter, we are now interested in analyzing the internal behavior of particle swarm optimization from the point of view of the set of informant particles that take part in its learning optimization procedure. In contrast to most studies in which only the results are researched, we here try to see what is happening inside the algorithm itself, why it works and how is the information in the informants of the velocity equation being actually used. After this understanding we will go for a proposal of a new PSO algorithm that outperforms the present state of the art in continuous benchmarking. As the starting point, we have based on the first analysis in the study presented in [CK02], where Clerc's constriction coefficient χ is proposed to be used in velocity update calculation instead of inertia weight, as shown in Equation 7.1.

$$\mathbf{v}_i^{t+1} \leftarrow \chi \left(\mathbf{v}_i^t + U^t[0, \varphi_1] \cdot (\mathbf{p}_i^t - \mathbf{x}_i^t) + U^t[0, \varphi_2] \cdot (\mathbf{b}_i^t - \mathbf{x}_i^t) \right) \tag{7.1}$$

$$\chi = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|}, \; with \; \varphi = \sum_i \varphi_i, \; and \; \varphi > 4 \tag{7.2}$$

Constriction coefficient χ is calculated, by means of Equation 7.2, from the two acceleration coefficients φ_1 and φ_2, being the sum of these two coefficients what determines the χ to use. Usually, $\varphi_1 = \varphi_2 = 2.05$, giving as results $\varphi = 4.1$, and $\chi = 0.7298$ [ES00, Tre03]. As stated by Mendes et all. [MKN04, MMWP05], this fact implies that the particle's velocity can be adjusted by any number of informant terms, as long as acceleration coefficients sum to an appropriate value, since important information given by other neighbors about the search space may be neglected through overemphasis on the single best neighbor. With this assumption, Mendes et all. [MKN04] proposed the Fully Informed Particle Swarm (FIPS), in which a particle uses information from all its topological neighbors. In FIPS, the value φ, that is, the sum of the acceleration coefficients, is equally distributed among all the neighbors of a particle. Therefore, for a given particle i with position $\mathbf{x}_i$, φ is broken up in several smaller coefficients $\varphi_j = \varphi/|\mathcal{N}_i|, \forall j \in \mathcal{N}_i$. Then, the velocity is updated as follows:

$$\mathbf{v}_i^{t+1} \leftarrow \chi \left[\mathbf{v}_i^t + \sum_{j \in \mathcal{N}_i} U^t \left[0, \varphi_j\right] \cdot (\mathbf{p}_j^t - \mathbf{x}_i^t) \right], \tag{7.3}$$

where $\mathcal{N}_i$ is the set of neighbors of the particle i, and following the neighborhood a given topology. Figure 7.1 illustrates the topologies used by Mendes et al. [MKN04] as the ones with most successful performances in a previous work [KM02]. These topologies are: All, Ring, Square, Four-Clusters, and Pyramid. Their results show that the Square topology (with 4 informants) outperforms the other ones. Indeed, the fact of defining these neighborhoods in the swarm makes the particles to be influenced only by a certain number of neighbors, and connected with static links in the graph. Once again, important information may be disregarded through overemphasis, in this case, on the structured sets of neighbors. The number of informants seems to play also an important role, but with no clue on how many of them is the best choice, or if even key issue for a higher performance is the topology itself or the fact that only a few informants are used.

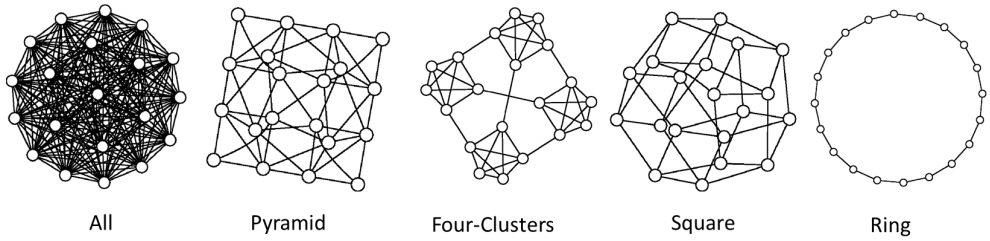


Figure 7.1: Topologies used by Mendes et al. [MKN04]. Each particle has a number of fixed neighbors in the swarm (All=N-1; Pyramid=3,5,6; Four-Clusters=4,5; Square=4; Ring=2)

All this motivated us to generalize the number of neighbors that influence particles, as well as the different configurations of topologies, in order to discover whether there exists a quasi-optimal number of informants that take part in the calculation of the velocity vector for a particular problem. Then, our initial hypothesis is that: **certain numbers (sets) of informant neighbors may provide new essential information about the search process, hence leading the PSO to perform more accurately than existing versions of this algorithm**, for a number of well-known benchmark problems in continuous optimization.

With the aim of researching in this line, we have designed in this work a generalized version of PSO that follows the information scheme of FIPS (with Clerc's constriction coefficient), but having as a free variable the number of informants in the calculation of the velocity vector. To evaluate our PSO with all possible configurations we have followed the experimental framework (with 25 problem functions) proposed in the Special Session of Continuous Optimization of CEC'05 [SHL$^+$05]. The performed analysis and comparisons (against Standard PSO and FIPS versions) will help us to claim if there are informant sets other than 2 and N that yield a more efficient PSO.

The remainder of this chapter is organized as follows. Next section presents the "Quasi-optimally Informed" version of PSO worked here. Section 7.3 describes the experimentation procedure and the parameter settings. In Section 7.4, experimental results are reported with analysis and discussions. After a series of initial considerations, Section 7.5 is devoted to conduct a through analysis from the point of view of the evolvability. Section 7.6 presents our proposal consisting on a hybrid PSO with quasy-optimal number of informants and with local search procedure. Finally, concluding remarks and future work are given in Section 7.7.

7.2 The Quest for an Optimal Number of Informants

As previously commented, the possibility of adjusting the particle's velocity by an arbitrary number of terms enables us to generalize the number (k) of neighbors, from 1 to Ss (being Ss the swarm size). Therefore, a number Ss of different versions of PSO can be generated (selecting k particles of the swarm without replacement), each one of them with neighborhoods containing k particles. Obviously, if $k = Ss$ the resultant version is the FIPS algorithm with neighborhood "ALL", as illustrated in Figure 7.1.

Nevertheless, since providing each k neighborhood with structured topologies is impracticable due to the great number of graph combinations, we have opted in this work to simply select k random (uniform) informants of the swarm (S). This way, for each particle i, and at each time step t, a different neighborhood ($\mathcal{N}_i^t$) with k elements is generated, and hence, the number of informants can be analyzed with independence of any structured topology. Formally, we can represent a given neighborhood as follows:

$$\mathcal{N}_i^t = \{n_1, \ldots, n_k\} \mid \mathcal{N}_i^t \subset S^t, \forall n_j, n_h \in \mathcal{N}_i^t, n_h \neq n_j \neq i \tag{7.4}$$

Following this scheme, our new PSOk performs as formulated in Equation 7.3, and using sets of k random (uniform) informant particles as neighborhoods. Then, we can evaluate all the PSOk versions (with $k : 1 \ldots Ss$) in order to discover whether an optimal value, or range of values, exist that allows to improve over the standard PSO and avoid the overhead of using topologies or computing contributions from all particles in the swarm.

Algorithm 8 Pseudocode of PSOk

```
1: φ_j ← φ/k
2: S ←initializeSwarm(Ss) /* Swarm S^0 with Ss number of particles */
3: while t < MAXIMUM_t do
4:    for each particle i of the swarm S do
5:       N_i^t ← generate_neighborhood(k, i, S^t) //Equation 7.4
6:       v_i^{t+1} ← update_velocity(v_i^t, x_i^t, φ_j, N_i^t) //Equation 7.3
7:       x_i^{t+1} ← update_possiton(x_i^t, v_i^{t+1}) //Equation 3.1
8:       p_i^{t+1} ← update_local_best(p_i^t, x_i^{t+1})
9:    end for
10: end while
```

The pseudocode of PSOk is introduced in Algorithm 8. After swarm initialization and φ_j value calculation (lines 1 to 3), the optimization process is repeated until reaching the stop condition. In this, at each iteration and for each particle a new neighborhood is randomly (uniformly) generated by fulfilling conditions of Equation 7.4 (line 6). Then, particle's velocity, current position, and local best position are updated (lines 7 to 9). Finally, the best so far particle position is returned as output.

7.3 Experimental Setup

In this section, we present an initial experimental methodology and statistical procedure applied to evaluate the different versions of PSOk and to compare them. We have followed the experimental framework presented in the Special Session on *Real-Parameter Optimization at CEC'05* [SHL$^+$05].

Table 7.1: Parameter setting used in PSOk

Description	Parameter	Value
Swarm size	Ss	30
Acceleration coefficient	φ	4.1
Constriction coefficient	χ	0.7298

We have implemented our PSOk using the MALLBA library [ALGN+07] in C++, a framework of metaheuristics. Following the specifications proposed in CEC'05 experimental procedure, we have performed 25 independent runs of PSOk for each test function and for each $k \in \{1, \ldots, Ss\}$ neighborhood. We use this standard benchmark to avoid biasing the results to concrete functions, and to have a high number of test problems that endorse our claims. For simplicity, the study has been made with dimension $D = 30$ (number of continuous variables), although further analyses with different problem dimensions are also included in sections 7.4.5 and 7.6. In the results, we are reporting the Maximum, the Median, the Minimum, and the Mean error of the best solutions found in the 25 independent runs. For a solution $\mathbf{x}$, the error measure is defined as: $f(\mathbf{x}) - f^*$, where f^* is the optimum fitness of the function. The maximum number of fitness evaluations has been set to $10,000 \times D$, which constitutes the stop condition.

To analyze the results, we have used non-parametric statistical tests, since several times the distributions of results did not follow the conditions of normality and homoskedasticity [GMLH09]. Therefore, the Median error (and not the Mean error), out of 25 independent runs, has been used for analysis and comparisons. In particular, we have considered the application of the Friedman's ranking test, and use the Holm's multicompare test as post-hoc procedure [She07].

The test suite of the CEC'05 benchmark is composed by 25 functions with different features [SHL+05]: unimodal, multimodal, separable, non-separable, shifted, rotated, and hybrid composed. Functions f1 to f5 are unimodal, functions f6 to f12 are basic multimodal, functions f13 and f14 are expanded, and functions f15 to f25 are composed by several basic functions. This way, our new proposals are evaluated under quite different conditions of modality, separability, and composition. Table 3.2 in Chapter 3 shows the function names, shape characterizations, bounds, and optimum values.

The parameter setting applied to PSOk (in Table 7.1) follows the specification of the Standard PSO in [PCG11]. The swarm size has been set to 30 particles in order to simplify the experimentation procedure due to space constraints. Nevertheless, as done with the problem dimension, additional experiments concerning different swarm sizes will be also provided in Section 7.4.4.

7.4 Analysis and Discussion

In this section, we present an analysis concerning the influence of the different neighborhood sizes (k) in PSOk. First, we will present a clear range for the informant number to be used, later we evaluate them against standard algorithms in the literature. Additional analyses relating the computational effort, the swarm size, and the problem dimension are also performed in this section.

7.4.1 Impact of the Number of Informants

First, we focus on the different number of informants constituting all possible combinations of neighborhoods in PSOk. Then, we show here the broad impact of this study aimed at discover the most promising values of k, with their possible implications in further studies about the PSO.

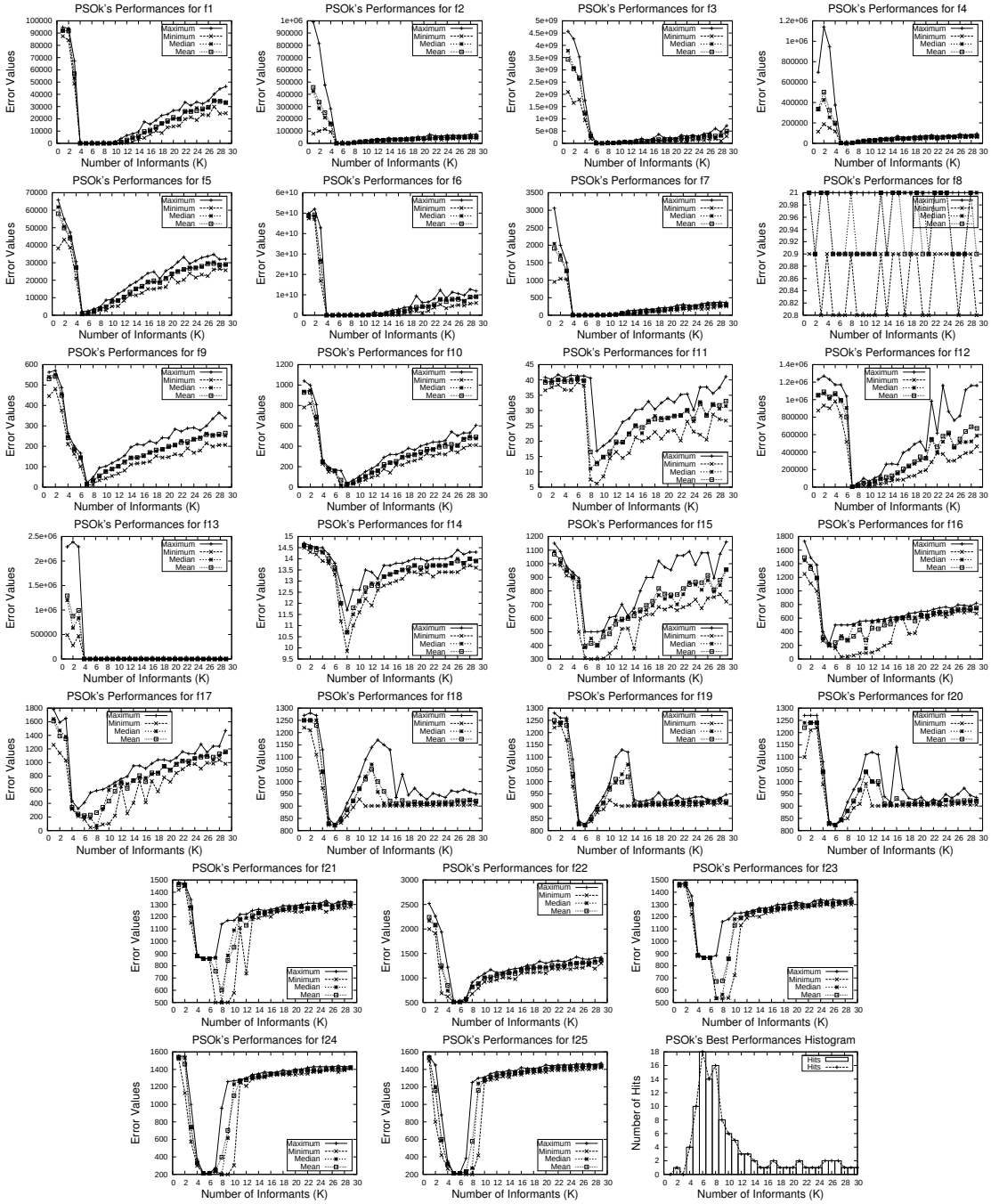


Figure 7.2: Each plot contains the performance (Maximum, Median, Minimum, and Mean error values out of 25 independent runs) of the different PSOk versions for the 30 possible values of k, and for all CEC'05 functions. The graph in the bottom-right figure contains the frequency histogram of best performance (number of Hits)

Since in this experimentation we have concentrated on a swarm size with 30 particles (later, we will analyze the influence of different swarm sizes), the number of PSOk's versions is 30, from PSO1 to PSO29, plus PSO30 represented by the so called FIPS-All. Therefore, we have undergone the evaluation of each version PSOk with the benchmark of functions CEC'05. Summing up, 25 independent runs for each algorithm version and for each function have been performed, resulting in a total number of $25 \times 25 \times 30 = 18,750$ experiments. The results are plotted in Figure 7.2, and several observations can be made from it:

- A number of 6 informants in the neighborhood makes the algorithm to perform with success in practically all functions. This is quite interesting since we can then propose the version PSO6 as the most promising one, and study its main features with regards to other parameters (swarm size, φ) and versus other algorithms (Standard PSOs, FIPS, etc.) in the next sections.

- For almost all functions, the interval between 5 and 10 informants concentrates most of the successful runs. In this sense, the plot at bottom-right in Figure 7.2 shows the histogram concerning the frequency in which each PSOk obtained the best results in the studied functions. This leads us to suspect that less than 5 informants is a deficient value of k not really taking particles out of the found local optima during the evolution, while more than 10 informants is redundant.

- In this sense, a number of 8 informants is also appropriate showing good performances in efficacy, although it is costly compared to PSO6. A new research question then comes to scene: could we create still better PSO's by using a range of informants during the search instead of betting for just one single constant value? A good hypothesis is that combining 6 and 8 informants in neighborhoods could be a source of new competitive algorithms.

- Another interesting observation concerns the behavior of all PSOk's versions in certain sets of functions that show similar curves of performance. Thus, functions f1 to f5, unimodal ones, show accurate performances from $k = 5$ in advance. Rastrigin's functions f9 and f10 draw quite similar curves, reaching their best performances with $k = 7$. Hybrid composed functions, from f15 to f25, show also high performance for $k = 6$.

- Curiously, biased functions to the same optimum f^* share similar curve shapes of PSOk's performances. For example, functions f1 to f4, with $f^* = -450$, functions f9 and f10, with bias to -330, functions f15, f16, and f17 which are biased to 120, functions f18, f19, and f20 with $f^* = 10$, functions f21, f22, and f23 with $f^* = 360$, and specially functions f24 and f25 biased to 260, they all show close curve shapes in Figure 7.2. An intriguing question is whether the CEC'05 benchmark is having an unknown feature in the induced landscapes that makes a given kind of PSO to perform better than others. If we could find such feature in the landscape domain we could create good algorithms from the start for these and other problems.

7.4.2 Performance Comparisons

We compare here the best PSOk version (PSO6) against the Standard PSO and other successful versions of FIPS with the aim of studying how well informed our proposal is.

Additionally, we have developed two simple combinations of PSOks with neighborhoods of 6 and 8 informants, namely PSO-U[6,8] and PSO-HE{6,8}. The former randomly (uniform) chooses

Table 7.2: Median error values for the 6 compared algorithms and for all the CEC'05 functions

Alg./Func.	Standard PSO 2007	FIPS-ALL	FIPS-Usquare	PSO6	PSO-U[6,8]	PSO-HE{6,8}
f1	5.68E-14	4.12E+04	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f2	5.08E+00	5.83E+04	3.38E-09	**3.41E-13**	1.54E+02	2.89E-05
f3	4.28E+07	5.39E+08	**6.36E+05**	2.06E+06	1.14E+06	3.02E+06
f4	5.05E+03	7.46E+04	1.50E+04	**6.30E-03**	4.03E+03	1.33E+00
f5	2.89E+03	3.09E+04	3.34E+03	1.25E+03	2.42E+03	**1.21E+03**
f6	1.66E+01	1.22E+10	1**.62E+01**	**1.62E+01**	2.70E+01	2.19E+01
f7	1.23E-02	3.33E+02	9.86E-03	**5.68E-14**	7.76E-03	**5.68E-14**
f8	**2.09E+01**	2.10E+01	2.10E+01	2.10E+01	2.10E+01	2.10E+01
f9	2.39E+01	2.73E+02	1.69E+01	1.51E+02	1.49E+01	**1.09E+01**
f10	1.80E+02	4.82E+02	**2.79E+01**	1.60E+02	1.44E+02	1.51E+02
f11	3.78E+01	**3.45E+01**	3.89E+01	3.97E+01	3.99E+01	3.98E+01
f12	2.88E+05	7.78E+05	**2.12E+03**	7.78E+05	4.61E+03	7.40E+05
f13	1.19E+01	6.67E+01	**2.99E+00**	1.37E+01	3.79E+00	1.19E+01
f14	1.40E+01	1.39E+01	1.28E+01	1.36E+01	**1.16E+01**	1.34E+01
f15	5.58E+02	9.46E+02	**2.39E+02**	3.57E+02	3.16E+02	3.06E+02
f16	2.12E+02	8.22E+02	**4.88E+01**	1.87E+02	1.69E+02	1.75E+02
f17	2.51E+02	1.19E+03	**7.24E+01**	2.01E+02	1.88E+02	1.90E+02
f18	8.30E+02	9.22E+02	8.31E+02	8.23E+02	8.42E+02	**8.22E+02**
f19	8.30E+02	9.33E+02	8.31E+02	**8.22E+02**	8.41E+02	**8.22E+02**
f20	8.30E+02	9.22E+02	8.30E+02	**8.23E+02**	8.41E+02	**8.23E+02**
f21	8.00E+02	1.32E+03	8.63E+02	8.58E+02	**6.80E+02**	8.58E+02
f22	5.23E+02	1.39E+03	5.51E+02	5.12E+02	5.74E+02	**5.11E+02**
f23	8.67E+02	1.34E+03	8.70E+02	8.66E+02	**5.54E+02**	8.66E+02
f24	2.16E+02	1.42E+03	2.21E+02	**2.12E+02**	2.30E+02	**2.12E+02**
f25	2.16E+02	1.44E+03	2.21E+02	**2.12E+02**	2.31E+02	**2.12E+02**
Hits	1	1	9	9	4	**10**

a value in {6, 7, 8} as the number of informants to be used in every step of the optimization process. The later version, PSO-HE{6,8}, performs the first half of the optimization process with 6 informants, and the remaining second half with 8 (Half Evolution, HE).

Table 7.2 shows the resulted median errors of compared PSOk versions for all CEC'05 functions. In addition, Standard PSO 2007, FIPS-ALL, and FIPS-USquare algorithms are also compared. We have added the FIPS-USquare (with informants in a square neighborhood) to this comparison since it was the version of FIPS that reported the best results in terms of performance in Mendes et al. [MKN04]. In this table, the best resulted median errors are marked in bold, and the last row summarizes the number of best results (Hits) obtained by each algorithm. As clearly observable, PSO-HE{6,8} obtains the higher number of Hits (10 out of 25), followed by PSO6 and FIPS-USquare with 9. In the case of PSO-U[6,8], a limited number of Hits of 4 leads us to suspect that the random combination of neighborhood sizes in the interval [6,8] does not make the most of these values. In general, we can also notice that all the algorithms obtain the best median errors for one function, at least, so even the Standard PSO 2007 in f8 and the FIPS-ALL in f11 report the best median error.

Statistically, Table 7.3 contains the results of an Average Rankings Friedman's test [She07] applied to the median results[1] of Table 7.2. We can see that PSO-HE{6,8} is the best ranked algorithm (with 2.58), followed by PSO6, and FIPS-Usquare. In contrast, FIPS-ALL is the worst ranked algorithm according to this test. This means that the complete scheme of information adopted in FIPS-ALL could damage the generation of new particles by incorporating noise and redundant information to them. In this sense, the FIPS-ALL shows even worse ranking than the Standard PSO 2007, whose set of informants (SI) is included in the set of the ALL topology, that is, $SI \subset ALL$. More precisely, in the same table, the adjusted p-values of a multicomparison

[1]Friedman statistic considering reduction performance (distributed according to chi-square with 5 degrees of freedom: 45.93714285714339).

Table 7.3: Average Friedman's Rankings with Holm's correction ($\alpha = 0.05$) of resulted median errors for CEC'05 functions

Algorithm	Rank	Holm's$_p$
PSO-HE{6,8}	2.58	-
PSO6	2.86	5.96E-01
FIPS-Usquare	2.88	5.70E-01
PSO-U[6,8]	3.26	1.98E-01
Standard PSO 2007	3.76	1.23E-02
FIPS-ALL	5.66	5.86E-09

Holm's test [She07] on the median errors are also shown. In this, the best ranked technique in the Friedman test, PSO-HE{6,8} is compared against all other algorithms. The Holm's procedure rejects those hypotheses of equality of distributions that have a p-value$\leq$0.05. Then, we can state that, for the tackled benchmark of functions (CEC'05), and according to this test, PSO-HE{6,8} is statistically better than Standard PSO 2007 (p-value=1.23E-02) and than FIPS-ALL (p-value=5.86E-09) algorithms.

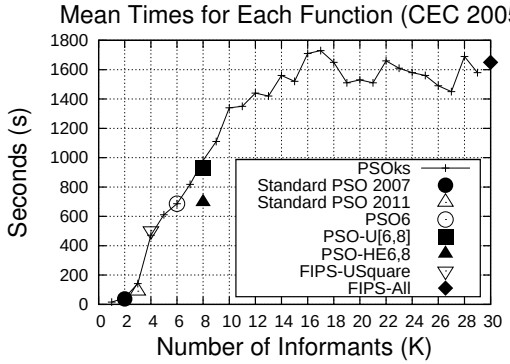


Figure 7.3: Mean running time (seconds) in which all PSOk versions have found the best mean error for all functions. The mean running time used by the rest of algorithms is also plotted

7.4.3 Computational Effort

We present in this section some remarks about the computational effort. To execute the experiments, we have used the computers of the laboratories of the Department of Computer Science of the University of Málaga (Spain). Most of them are equipped with modern dual core processors, 1GB RAM, and Linux S.O., having into account that there are more than 200 computers, that means that up to 400 cores have been available. To run all the programs, we have used the Condor [TTL05] middleware that acts as a distributed task scheduler (each task dealing with one independent run of PSOk).

Figure 7.3 plots the mean running time (seconds) in which all the versions of PSOk have found the best mean error for all functions. The mean running times used by PSO-U[6,8], PSO-HE{6,8}, Standard PSO 2007, FIPS-ALL, and FIPS-USquare algorithms are also plotted. As expected, the running time increases with the number of informants in PSOk, although it seems to stabilize from

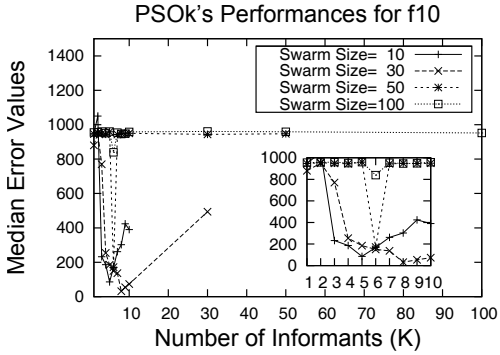


Figure 7.4: Influence of the different swarm sizes in PSOk. The median fitness values are plotted for swarm sizes with 10, 30, 50, and 100

PSO15 to PSO30 (FIPS-ALL). We have to mention that this last version (PSO30) does not use the random selection of informants, since all particles in the swarm are involved in the velocity calculation. This led us to suspect that the time the random selection of informants spends is not significant with regards to the time of calculating the new velocity vector (information time).

We can also observe in this figure that the Standard PSO 2007 required the shortest running time (excepting PSO1 and PSO2), since only two informants are involved in the velocity calculation: the personal ($\mathbf{p}$) and the global best ($\mathbf{b}$) positions. Almost all the remaining compared algorithms required similar running times, between 600 and 900 seconds, since they used a close number of informants (from 4 to 8) in their operations. Obviously, the algorithm using the higher number of informants, PSO30 (FIPS-ALL), required the longest running time.

7.4.4 Influence of the Swarm Size

Another interesting feature of PSOk that we also analyze here concerns the influence that the swarm size experts on the optimal number of informants k in the neighborhood. In this sense, we have carried out a series of additional experiments in which, four configurations of swarm sizes (with 10, 30, 50, and 100 particles) have been set in the different PSOk algorithms for a number of neighborhoods (k). Specifically, we have evaluated from PSO1 to PSO10, PSO30, PSO50, and PSO100 versions, each one of them with the four possible configurations of swarm size.

Figure 7.4 shows the plot of the median error values resulted from the experiments with different sizes of swarm, for function f10 (of CEC'05). We have selected this function since it shows a representative behavior similar to the ones obtained on the remaining functions. We can effectively observe that all the best median errors are obtained by PSOk versions with neighborhoods included in the range of $k \in [5, 9]$. Therefore, with independence of the number of particles in the swarm, the empirical optimal number of informants required is included in this interval, and even for larger swarm sizes (with 100 particles) the performance of PSOk is enhanced using those small neighborhoods.

7.4.5 Influence of the Problem Dimension

Similar to the previous analysis, the potential influence that scaling to larger problem dimensions may have on the selection of the neighborhood size (in PSOk) is studied in this section. In this

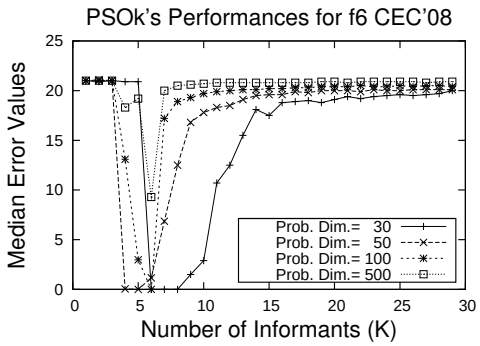


Figure 7.5: Influence of different problem dimensions in PSOk. Study made with the Shifted Ackley's function, f6 in CEC'08 and f3 in CEC'10

case, the experiments are focused on the resolution of large scale problems, as the ones found in the context of the Special Session CEC'08 [TYS+07], CEC'10 [TLS+10] and SOCO'10 [HLM10b] (the stop condition is $5,000 \cdot D$ fitness evaluations). In concrete, we have worked with the Shifted Ackley's function (f6 in CEC'08, SOCO'10 and f3 in CEC'10) with dimensions $D = 30$, 50, 100, and 500 variables. The swarm size was set to 30 particles as in initial experiments, and our goal is to see how sensible is PSOk to problem size.

Figure 7.5 plots the median errors resulted for all PSOk configurations. Once again, the informed PSOk with neighborhood sizes close to 6 informant particles show the best performance for almost all problem dimensions. Only when dealing with 50 variables, PSOk with 4 and 5 informants obtain the best median errors values, also close to PSO6. Therefore, as expected, increasing the number of variables in the problem dimension does not seem to variate the behavior of the PSOk versions. In fact, for the Shifted Ackley's function, the curve shapes representing each problem dimension follow similar patterns to practically all plots in Figure 7.2.

7.5 Evolvability Analysis

In this section, we go one step beyond in this research line by analyzing the internal behavior of PSO from the point of view of the evolvability. Our main motivation is to find evidences of why certain unstructured neighborhoods, with 6±2 informant particles, perform better than other neighborhood formulations.

With this aim, we have computed both, fitness and distance to optimum traces to calculate evolvability measures of PSOk with different combinations of informants. We have followed the experimentation framework of CEC'05. In fact, this test suite has been properly characterized in a recent work [MS11] by means of the Fitness Distance Correlation (fdc), so we have decided to use this measure in this work together with the Fitness Cloud (fc) and Escape Probability (ep), to test the resulting landscape characterizations from all combinations of neighbors in PSOk. With the outlined information, we expect to test the second work hypothesis: **a number of 6±2 informants in the operation of PSO may compute good particles for longer than other PSO formulations in multiple complex benchmarking problems**. We here test that, on the one hand, few informants (one or two as in Standard PSO) could sometimes show high escaping probabilities and positive correlation, although evolving solutions with poor fitness values. On the

other hand, we will show that in PSO with more than 10 informants, solutions could concentrate on small regions whose sizes decrease as the neighborhood topology increases, thus confusing the search and again reducing the escape probability

7.5.1 Evolvability Measures

Before we pass to discuss the results, we describe here the evolvability measures used in this work and their application to the particular case of PSO.

- *Fitness-distance* analysis quantifies the relation between the fitness of particles $f(\mathbf{x}_i)$ in the landscape and their distances to the nearest global optimum $\mathbf{x}_{min}$ (assuming that we are minimizing) [LLY11]. Distance between points in continuous domains are measured using Euclidean distance d_E. Then, the fitness-distance correlation (fdc) can be quantified by the r_{fdc} coefficient:

$$r_{fdc} = \frac{c_{fd}}{s_f \cdot s_d}, being \;\; c_{fd} = \frac{1}{n}\sum_{i=1}^{n}(f_i - \overline{f}) \cdot (d_i - \overline{d}) \tag{7.5}$$

where f_i is the fitness of solution $f(\mathbf{x}_i)$, d_i is the distance of the solution $\mathbf{x}_i$ to the optimum $d_i = d_e(\mathbf{x}_i, \mathbf{x}_{min})$, $\overline{f}$, $\overline{d}$, s_f, and s_d are the means and standard deviations of the fitness and distance samples, respectively. In our experiments, for each function and for each PSOk version, we identify the samples in 25 independent runs with the 1% best fitness values to compose the r_{fdc}. An interpretation of this coefficient can be as follows [MS11]: a positive r_{fdc} near to 1 means globally convex single-funnel topologies. A value around 0 of this coefficient may indicate plateau shape landscapes with tiny sharp basins and problems without any global structure. A negative value of r_{fdc} indicate the existence of "deceiving" landscapes, where an optimizer (PSO in our case) perceives poor objective function values closer to the minimum than farther away.

- *Fitness-fitness* or *Fitness Cloud* (fc) [VCC$^+$04] analysis is basically a plot of fitness values of individuals against fitness values of their neighbors. By definition of fc, for each sampled individual $\mathbf{x}_i$ with fitness f_i, generate k neighbors by applying a genetic operator to $\mathbf{x}_i$, and let be $\overline{f_i}$ the mean fitness of all neighbors of $\mathbf{x}_i$. Then, the set of points $\{(f_1, \overline{f_1}), \ldots, (f_n, \overline{f_n})\}$ is taken as the fitness cloud. In our particular case, we are interested in computing the fitness cloud by plotting the fitness (f'_i) of a new particle ($\mathbf{x}'_i$) that is generated from their neighbors (using velocity rule in Equation 3.11), and the mean fitness $\overline{f_i}$ of all these neighbors of $\mathbf{x}_i$. Now, the set of points $\{(f'_1, \overline{f_1}), \ldots, (f'_n, \overline{f_n})\}$ can be taken as the fitness cloud.

- The *Escape Probability* (ep) [Mer04] considers the number of steps required to escape from a local optimum. It is defined as $P(f_i) = \frac{1}{S_i}$, where S_i denotes the average number of steps required to find an improving move starting in an individual with fitness value f_i. In the context of our PSO, we averaged the improving intervals (evaluation steps) computed by the each particle of the swarm to calculate escape probability through the iteration process. In next section, several examples of ep are plotted for f24.

Since in this experimentation we have concentrated on a swarm size with 30 particles, the number of PSOk's versions is 30, from PSO1 to PSO29, plus PSO30 represented by the so called FIPS-All. Summing up, 25 independent runs for each algorithm version and for each function have been performed, resulting in a total number of $25 \times 25 \times 30 = 18,750$ experiments. The resulting r_{fdc} coefficients are plotted in Figure 7.6 as bar graphs. Next, we make several observations about this figure. Later, we will focus on interesting observations concerning fc and ep.

Figure 7.6: Each plot contains the mean r_{fdc} coefficients in the Y-axis (out of 25 independent runs) of the different PSOk versions (for the 30 possible values of k) in X-axis, for all CEC'05 functions

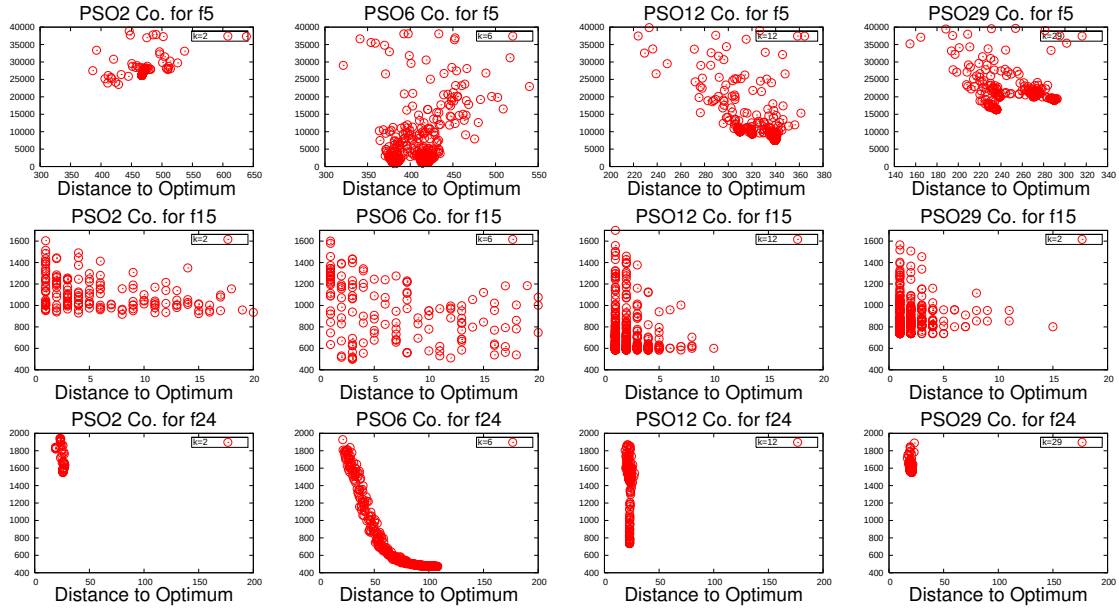


Figure 7.7: Fitness-distance plots of functions f5, f15, and f24, generated from independent executions of PSO with 2, 6, 12, and 29 informants

7.5.2 Fitness-Distance Analysis

In general, we can first observe from Figure 7.6 that PSO with 6 informants shows $r_{fdc} >> 0$ in single-funnel functions (f1-f10) and $r_{fdc} << 0$ in multi-funnel ones (f11-f25). This means that using 6 informants, our proposal is able to distinguish between globally convex and/or deceiving (multiple convexities) landscapes, and independently to the problem modality. Nevertheless, three exceptions are registered: for f8, with plateau landscape and hence, highly dependent to the swarm initialization, and for f16 and f17, which were characterized as bi-funnel in [MS11]. For these two last functions, there is a certain probability of evolving samples (from initialization) on the funnel where the optimum is located [SWLH06], which contributes to compute a r_{fdc} higher than 0.

Second, for a series of functions (f3, f6, f11, f12, f14, f14, and f25) there exists a change from negative to positive in the tendency of r_{fdc}, resulted from PSOk versions with number of informants close to 6. This suggests that using few informants (<4) could create a tendency to quickly move toward non interesting regions, leading the algorithm to show an early stagnation and obtaining weakly correlated solutions far from the optimum.

More in detail, Figure 7.7 shows fitness-distance plots resulted from independent executions of PSO with 2, 6, 12, and 29 informants, for functions f5, f15, and f24. We can observe that using few informants (2 in this case), the algorithm shows correlation for the three functions, but evolving solutions with poor fitness values. With more than 10 informants, solutions are again correlated, although concentrating on small regions on multi-funnel landscapes (as plotted for f15 and f24). Using 6 informants in PSO is the best trade-off between fitness-distance correlation and fitness quality. A special case can be observed for f24 (highly deceptive), where solutions are gradually improving although showing certain distance with regards to the global optimum.

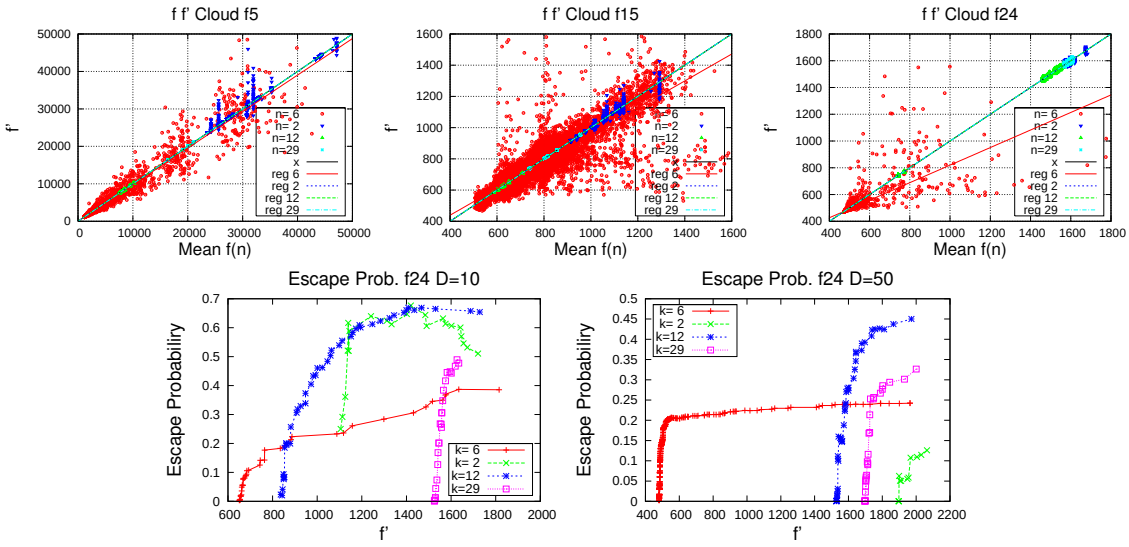


Figure 7.8: Fitness-fitness clouds $\{f', \overline{f(n)}\}$ of functions f5, f15, and f24, generated from independent executions of PSOk with 2, 6, 12, and 24 informants. Graphs in bottom plot the Escape Probability concerning f24 for dimensions 10 and 50

7.5.3 Fitness-Fitness Analysis

Figure 7.8 plots the Fitness Clouds of functions f5, f15, and f24, generated from independent executions of PSO with 2, 6, 12, and 29 informants. For the three functions, a number of 6 informants is able to keep for longer the generation of new better particles with improving fitness (f'). In fact, regression lines calculated from distribution clouds (of 2, 12, and 29 informants) generally fit the diagonal line of plots, whereas regression lines of 6 informants' clouds show positive slopes in final evolution steps, with regard to diagonal lines. This means that new fitness values f' calculated using 6 informant neighbors improve more frequently the mean fitness $\overline{f(n)}$ of these neighbors.

An illustrative example of this behavior can be explicitly observed in Figure 7.8 bottom, where the Escape Probability concerning f24 is progressively plotted. Solutions evolved by PSO6 generally show a moderate *ep* progress, although reaching a deeper basin of local optimum, e.g., better fitness values.

After this comprehensive experimentation, we have tested our initial research question showing that 6 informants in PSO is the best trade-off between fitness-distance and fitness quality. This is noticeable since the algorithm is simple (Occams razor), and these results together with the previous ones in Section 7.4 suggest that this number of 6±2 informants put into PSO a better learning procedure than other combinations of informants, which makes it able to generate good particles for a longer time.

7.6 PSO6 with Multiple Trajectory Search

In spite of the advantage of observed in PSO with 6±2 informants, the new proposal (PSO6) may still show a certain local basin attraction and moderate performance on non-separable complex problems, as typically observed in particle swarm versions. Therefore, additional mechanisms have to be used to correct this drawback.

In this section, we add to our PSO6 a local search method, with the aim of improving its performance on complex problems with different landscape features: multimodal, multifunnel, non-separable, shifted, and rotated. First, as proven in [SWLH06], PSO tends to converge quickly to the local basin that contains the majority of particles at initialization. A consequence of this observation is that PSO may exhibit higher tendency to stagnate on multimodal functions than other algorithms. In this case, we use our PSOk with six informants that performs an optimized learning procedure to move particles to more interesting regions, and hence avoid the attraction to non promising local basins.

Second, as most of PSO versions work dimension by dimension, it is actually hard to find the problem optima when it is located far from the origin (or when it is on an axis or a diagonal) of coordinates in non-separable problems. To alleviate this last issue, we incorporate a local search method to our PSO6 by means of which, particles move individually, exploring their problem neighborhoods in the context of variations of dependent variables.

Our new proposal, called PSO6-Mtsls (PSO6 with Multiple Trajectory Search), is then evaluated on a set of 40 benchmark functions in order to validate whether it is competitive with the current state of the art in continuous optimization, or not.

7.6.1 The Proposal: PSO6-Mtsls

Our proposal, PSO6 with Multiple Trajectory Search (PSO6-Mtsls), consists in running PSO6 as a baseline method in which we have incorporated a local search mechanism to improve solutions obtained by the particle swarm algorithm. In concrete, we have employed the well-known LS1 [TC08] of MTS because of three main reasons: (1) LS1 is the responsible of most of the MTS performance, (2) it has been proven to be an efficient optimizer on large scale and non-separable complex problems [TC08], and (3) it has been successfully used to hybridize other swarm intelligence approaches like IPSO [MdOSVdED11], ACO [LMdOA$^+$11], and DE [MLTPn09].

In the context of the collective learning procedure induced by informant particles in PSOk, the LS1 procedure can be interpreted as a particle's individual learning ability that allows it to explore-explode its immediate area neighborhood in the absence of any social influence. In this sense, the movement of an individual particle depends on the improvement obtained from variations in its adjacent variables (solution dimensions) and hence, the interdependency of variables in non-separable problems can be tackled more effectively than simply using vector operators inducing linear combinations (as in PSO versions).

PSO6-Mtsls invokes a local search routine after a certain number of iterations, performing successive improvements on the global best particle ($\mathbf{b}^t$) obtained by PSO6 at moment (t) of invocation. We have to notice that, in spite of PSO6 does not work directly with the global best particle (but indirectly if it happen to take part in the current set of six informants), it is kept updated through the iteration procedure, to be used by the local search LS1 as a target particle. If PSO6-Mtsls detects that an application of LS1 does not improve the solution, the local search is stopped. In this way, the additional cost in terms of extra function evaluations performed by LS1 can be lighten.

Algorithm 9 Pseudocode of PSO6-Mtsls

```
1: φ_j ← φ/k
2: S ←initializeSwarm(S) /* Swarm S with Ss number of particles */
3: /************* Initializing Search Subranges *************/
4: for k = 1 to Ss do
5:    Improve_k ← True
6:    SR_k ← (Upper_Bound − Lower_Bound) · 0.5
7: end for
8: while t < MAXIMUM_t do
9:    /************* PSOk, with k = 6 *************/
10:   for each particle i of S do
11:      N_i^t ← generate_neighborhood(k, i, S^t)   //Equation 7.4
12:      v_i^{t+1} ← update_velocity(v_i^t, x_i^t, φ_j, N_i^t)   //Equation 3.11
13:      x_i^{t+1} ← update_positon(x_i^t, v_i^{t+1})   //Equation 3.1
14:      p_i^{t+1} ← update_local_best(p_i^t, x_i^{t+1})
15:   end for
16:   b^{t+1} ← update_global_best(b^t)
17:   /***************** MTS:LS1 *****************/
18:   if t%ls_freq = 0 then
19:      X_k ← b^{t+1}
20:      for j = 1 to #max_ls_iters do
21:         Improve_k, SR_k ← LS1(X_k, Improve_k, SR_k)
22:         if Improve_k = False then
23:            break
24:         end if
25:      end for
26:      update(t)
27:   end if
28: end while
```

The pseudocode of PSO6-Mtsls can be observed in Algorithm 9 and is organized as follows: the first phase, from line 1 to 7, corresponds to parameter setting and swarm initialization. For the initialization of particles (line 3), we have partially used the method proposed in [MPnLR10] to generate good diverse solutions. This method starts with the partition of the range of each dimension to *sr* subranges of equal size. Then, for each particle, a subrange for each dimension is selected based on the inverse probability of the frequency count associated with the subrange. Finally, a value is uniformly generated within the selected interval and the frequency count associated to the subrange is incremented.

In a second phase, the PSO6-Mtsls is then iterated until the stop condition is met: a given number of function evaluations is reached. Finally, the PSO6 performs one iteration (lines 10 to 15), the global best is updated in line 16, and the local search L1 is invoked with a certain frequency according to ls_freq (line 18 to 27). In this case, the local search procedure is repeated a given number of iterations max_ls_iters while the solution is successively improved. In other case, the local search is aborted and the PSO6 follows with a new iteration. Once the stop condition is reached, the algorithm returns the best particle found so far.

Table 7.4: Complete parameter settings

Parameter Value	Algorithms
Swarm size $Ss = D \cdot 0.7$	All
Acceleration coefficient $\varphi = 4.1$	PSO6, PSO6-Mtsls, FIPS-ALL, FIPS4
Acceleration coefficient $\varphi = 0.5 + \ln(2)$	S2011
Inertia weight $\omega = 1/(2 \cdot \ln(2))$	S2011
Constriction coefficient $\chi = 0.7298$	PSO6, PSO6-Mtsls, FIPS-ALL, FIPS4
Number of Informants $k = 6$	PSO6, PSO6-Mtsls
Number of Informants $k = 4$	FIPS4
Number of Informants $k = Ss$	FIPS-ALL
Number of Informed by a give one $k = 3$	S2011
Topology $T = U - Random$	PSO6, PSO6-Mtsls
Topology $T = Square$	FIPS4
Topology $T = Complete$	FIPS-ALL
Topology $T = U - Random$	S2011
Local search frequency $ls\_freq = 5$	PSO6-Mtsls
Max. ls. Iterations $max\_ls\_iters = 70$	PSO6-Mtsls

7.6.2 Experiments with PSO6-Mtsls

This new experimental study is structured in three different phases. First, we evaluate our PSO6-Mtsls and other related PSO versions concentrating on the original PSO6 without any local search procedure, the Standard PSO 2011 (S2011), the Fully Informed PSO (FIPS-ALL), and the Fully Informed Square Neighborhood (FIPS4, the best one in [MKN04]), by comparing their performances. Second, our proposal is compared against other 15 algorithms featured in SOCO'10, on different problem scales with dimensions 50, 100, 200, and 500 continuous variables. The third experimental phase corresponds to the evaluation of PSO6-Mtsls with regards to other similar modern swarm intelligent approaches also hybridized with local search methods: IPSO-Powell [MdOAS11], IPSO-Mtsls [MdOSVdED11], and IACOr-Mtsls [LMdOA$^+$11].

For the two first phases, we used the 19 functions (labeled *soco*$*$) provided in SOCO'10. In this benchmark, functions *soco*1 to *soco*6 were originally used in CEC'08 [TYS$^+$07]. Functions *soco*7 to *soco*11 were added to the first ones in the special session of ISDA'09 [HL09a], and functions *soco*12 to *soco*19 consist on hybridized functions that combine two others (being one of them non-separable). For the third phase, we extended the working set by including 21 more functions of CEC'05 (labeled as *cec*$*$) to the previous 19 of SOCO'10, then constituting a set of 40 functions. We have to notice that, as done in [LMdOA$^+$11], from the original 25 functions of CEC'05 we omitted *cec*1, *cec*2, *cec*6, and *cec*9, since they are the same as *soco*1, *soco*3, *soco*4, and *soco*8. Table 3.2 in Chapter 3 shows the set of functions used in this study with their most interesting features: unimodal, multimodal, separable, non-separable, shifted to biased optimum, rotated, and hybrid composed. The respective bounds of search ranges and biases to optima are also indicated. The detailed descriptions of all these functions can be found in [HLM10b] and [SHL$^+$05].

Following the specifications of the two benchmarks used, we have applied as stop conditions a maximum number of 5,000$\cdot D$ evaluations for SOCO'10, and 10,000$\cdot D$ evaluations for CEC'05 functions. We performed 25 independent runs for each investigated algorithm and problem dimension. We report the error values of the best solutions found defined as: $f(x) - f^*$, where f^* is the optimum of the function f. Error values lower than 10^{-14} (*0-threshold*) are approximated to zero.

Table 7.5: Swarm size parameter tuning: successful runs with best performances resulted by PSO6, with different swarm sizes on different problem dimensions

Dimension	Swarm Size				
	20	30	60	100	200
50	0	19	0	0	0
100	0	0	19	0	0
200	0	0	0	19	0
500	0	0	0	0	19
Diff.		+	+	+	+

Table 7.6: LS1 parameter tuning: best performed values are in bold

LS1 parameter	Values for problem dimension			
	50	100	200	500
ls_freq	5,10,**30**,50	**5**,10,30,50	**5**,10,30,50	**5**,10,30,50
max_ls_iters	10,30,**50**,70	10,30,50,**70**	10,30,50,**70**	10,30,50,**70**

The parameter setting applied to PSO6, as well as to the other evaluated PSO versions, are shown in Table 7.4 and follow the specification of their original works were they where proposed [GNA11a], [MKN04], and [PCG11]. Nevertheless, concerning the swarm size, we have decided to perform an additional parameter tuning with PSO6, since in this work we are using a large set of functions with different dimension scales. Therefore, we have carried out a preliminary experimentation with PSO6 by setting it with different combinations of swarm sizes and problem dimensions in the context of SOCO'10.

Table 7.5 contains the number of functions for which PSO6 obtains the best median results for each combination of swarm size and problem dimension. In this table, we can easily observe that the swarm size seems to be proportional to the problem dimension, since a higher swarm is more accurate for large scales, and opposite. The last row specifies that statistical differences were found in distributions (+). For this reason, and after additional runs, we have opted to use a linear proportion to set the swarm size by using the 70% of the problem dimension as the number of particles in the swarm. In this way, we use $Ss = D \cdot 0.7$ in Table 7.4 for PSO6, as well as for all other PSO versions.

In the case of our PSO6-Mtsls, specific parameters to the particle swarm use the same setting as in PSO6, including the proportional swarm size. For specific parameters to LS1, a series of tuning experiments have been also carried out to find an accurate combination of local search frequency and maximum number of iterations in the local search procedure. Table 7.6 shows the experimented values for LS1, where the best parameter combination is in boldface. As expected, the higher frequency and the maximum number of LS1 iterations shows the better performance for almost all problem dimensions. Only in the case of $D = 50$, a different combination performed better with $ls\_freq = 30$ and $max\_ls\_iters = 50$. We suspect that more frequent and large LS1 procedures in PSO6-Mtsls could be costly when $D = 50$, that is, the shorter scale in SOCO'10, for which a lower number of function evaluations are allowed. Nevertheless, for the sake of a homogeneous parameter setting, we have decided to use always the best combination for almost all problem dimensions: $ls\_freq = 5$ and $max\_ls\_iters = 70$. In fact, this combination is close to the ones used in related works in the literature [MdOSVdED11], [LMdOA+11], [TC09]. Parameters of all other compared algorithms can be found in their reference works.

Table 7.7: Average Friedman's rankings with Holm's correction ($\alpha = 0.05$) for SOCO'10 functions with dimensions 50 and 100

Algorithm	50		100	
	Rank	$Holm's_{p-value}$	Rank	$Holm's_{p-value}$
PSO6-Mtsls	**1.367**	-	**1.152**	-
PSO6	2.149	0.16E-01	2.331	4.36E-02
FIPS4	2.915	0.44E-02	2.952	2.05E-02
S2011	3.684	2.34E-04	3.763	6.55E-05
FIPS-ALL	5.000	2.90E-10	5.000	7.26E-11

7.6.3 PSO6-Mtsls: Performance Comparisons

This section is devoted to show all the performance results of our PSO6-Mtsls. A series of comparisons with other PSO versions, as well as with other modern proposals in the current state of the art are carried out from different points of view. Our goal is to solve different problems as well as highlighting its advantages.

Comparison of PSO Versions

Figures 7.9 and 7.10 show the boxplots representing que distributions of error fitness obtained by Standard PSO 2011 (S2011), Fully Informed PSO with complete neighborhood (FIPS-ALL), FIPS with Squared Neighborhood (FIPS4), PSO6, and our proposal here PSO6-Mtsls, for the 19 functions of SOCO'10 benchmark. Table 7.7 shows the average ranking of compared algorithms resulted from the Friedman's statistical test and applying a post-hoc Holm's correction for multiple comparisons ($\alpha = 0.05$), for problem dimensions 50 and 100. In this table, the algorithm with the best ranking is used as control method (marked in boldface). Thus, those algorithms with adjusted Holms's *p-values* < 0.05 are statistically outperformed by the control method.

In general, we can observe in figures 7.9 and 7.10 that PSO6-Mtsls shows the best performance in (almost) all functions, and for the two analyzed problem dimensions. As shown in Table 7.7, our proposal is the best ranked algorithm and therefore, it is set as control method for the post-hoc Holm's test. For dimension 50, all compared PSO versions excepting PSO6 are statistically outperformed by PSO6-Mtsls. We have to notice that LS1 parameters were set using a homogeneous tuning for all dimensions, although being slightly disadvantageous in the particular case of dimension 50. We suspect that using specific parameter setting for this dimension could lead our PSO6-Mtsls to be statistically better than PSO6. In fact, in the case of dimension 100, we can effectively observe that our proposal outperforms all other compared algorithms, including PSO6, with statistical confidence.

If we examine non composed functions (*soco*1 to *soco*11) in Figures 7.9 and 7.10, we can clearly observe that PSO6-Mtsls always shows the best results, followed by PSO6, FIPS4, S2011, and FIPS-ALL. Nevertheless, there are several functions: *soco*1, *soco*6, and *soco*7, for which PSO6-Mtsls obtained similar distributions to the ones of PSO6, and FIPS4. Not surprisingly, these functions are characterized as separable in SOCO'10, and hence, PSO6 and FIPS4 are also able to show accurate performances with regards to our proposal. Therefore, the possible benefits induced by the local search method could be said to pay in non-separable functions.

Concerning composed functions (*soco*12 to *soco*19), the error distributions obtained by PSO6-Mtsls are in general better than the ones of compared PSO versions. Therefore, we can claim that the use of local search (LS1) in PSO6 is advantageous in the context of SOCO'10 benchmark of

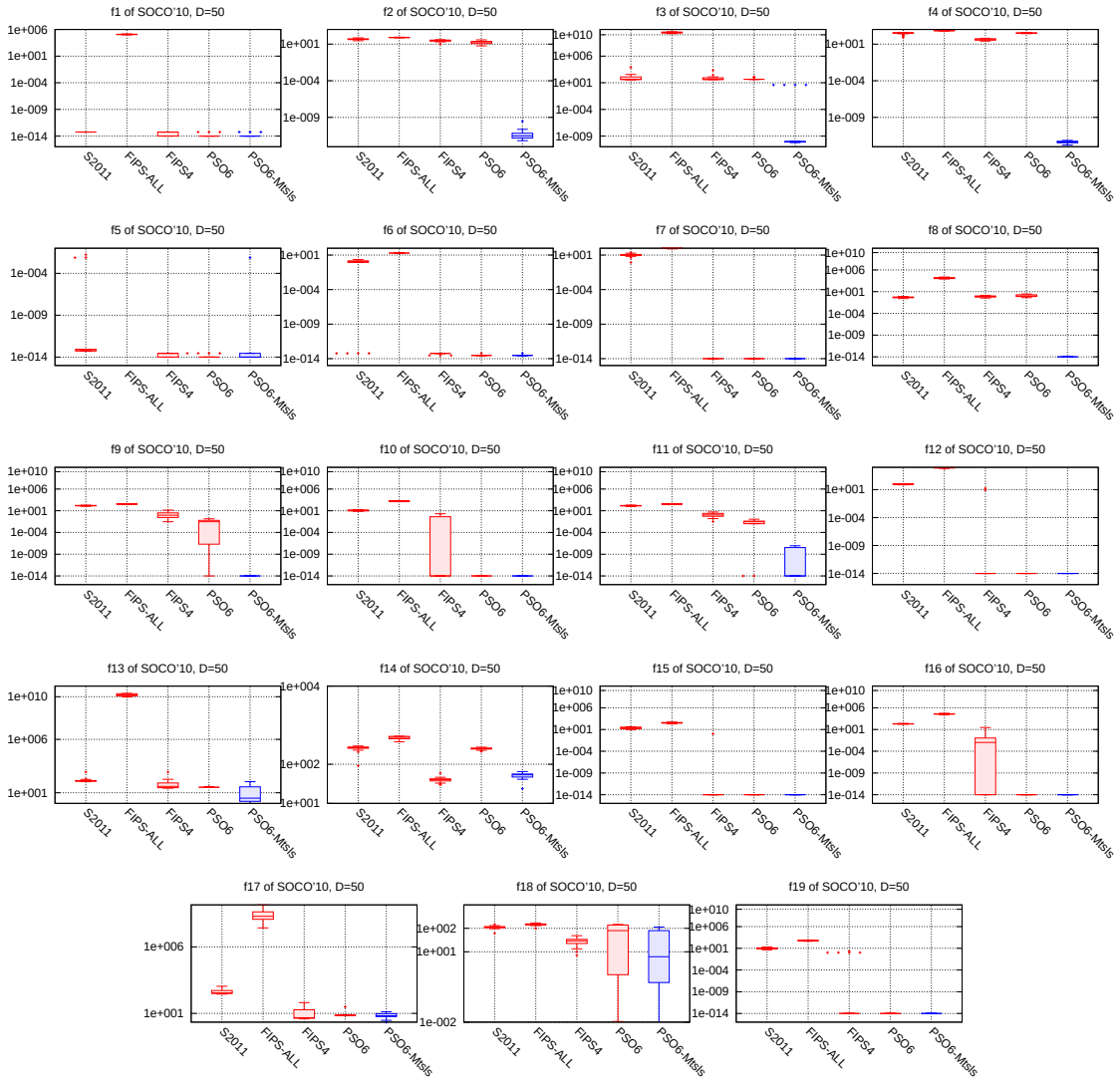


Figure 7.9: SOCO'10 function's fitness distributions of S2011, FIPS-ALL, FIPS4, PSO6, and PSO6-Mtsls, for dimension 50

functions. However, a single exception can be observed for function *soco*18 with dimension 100, where the high proportion of *soco*4 (separable) variables in the composition with *soco*9 is the probable reason for PSO6 to show a better error distribution, even without any local search procedure. In this sense, a secondary observation is that PSO6 generally shows a better performance than FIPS4 (the best algorithm in [MKN04]), and is statistically better than FIPS-ALL and S2011. This result was also founded in our previous work [GNA11a], although in the context of CEC'05 benchmark of functions, with dimension 30.

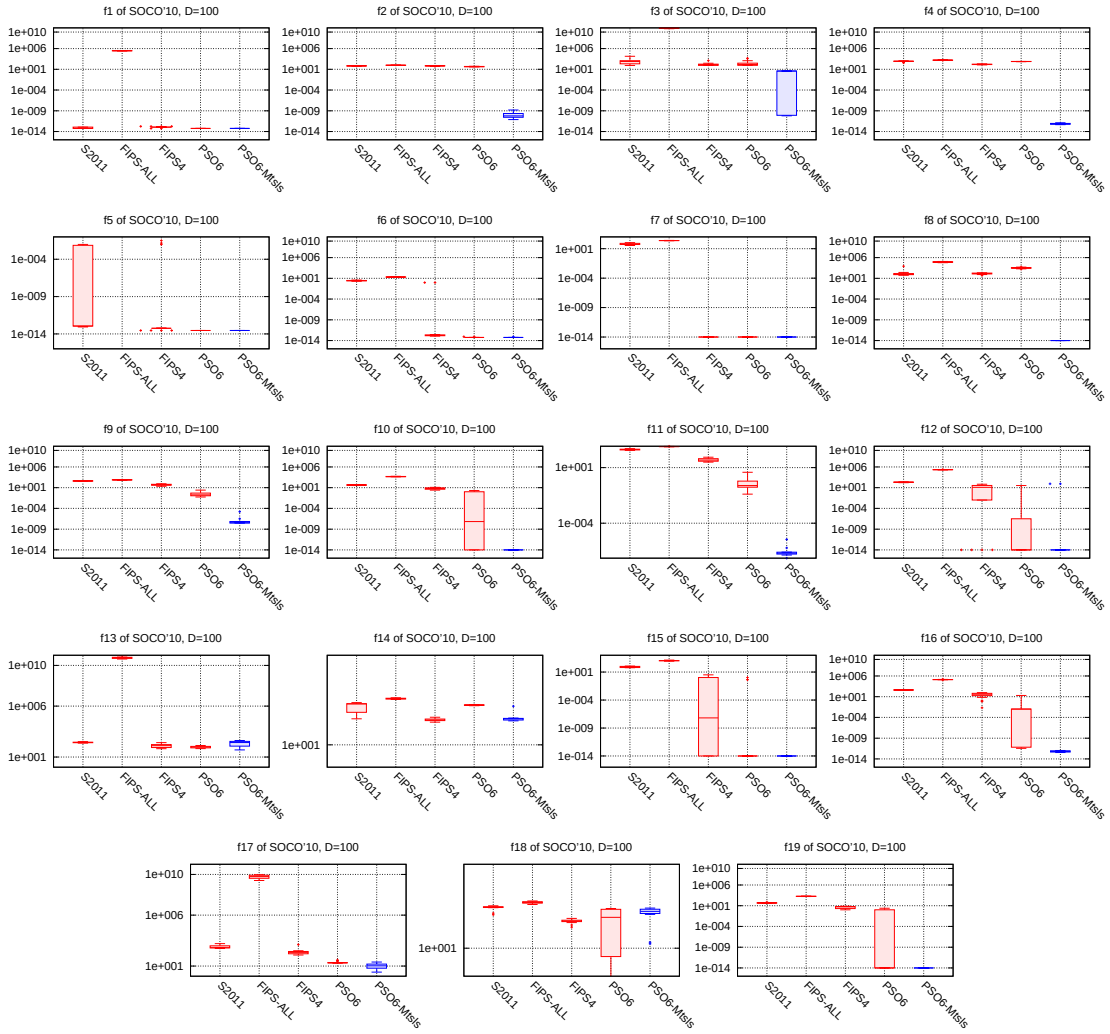


Figure 7.10: SOCO'10 function's fitness distributions of S2011, FIPS-ALL, FIPS4, PSO6, and PSO6-Mtsls, for dimension 100

Comparisons with Other Algorithms in the State of the Art

Figure 7.11 shows the boxplots representing the median error distributions of the 19 SOCO functions obtained with PSO6, PSO6-Mtsls, and the algorithms [2] featured in the special issue of SOCO'10, for dimensions 50, 100, 200, and 500. From these last algorithms, the results of DE, CHC, and G-CMA-ES were provided as base-reference techniques to compare with, previous to the global comparisons. In relation with this figure, Table 7.8 contains the results of applying the Friedman's test and Holm's corrections to the aforementioned distributions of median errors, for all compared algorithms and dimensions.

[2]The complete information about featured algorithms in SOCO'10 is available in `http://sci2s.ugr.es/EAMHCO/`

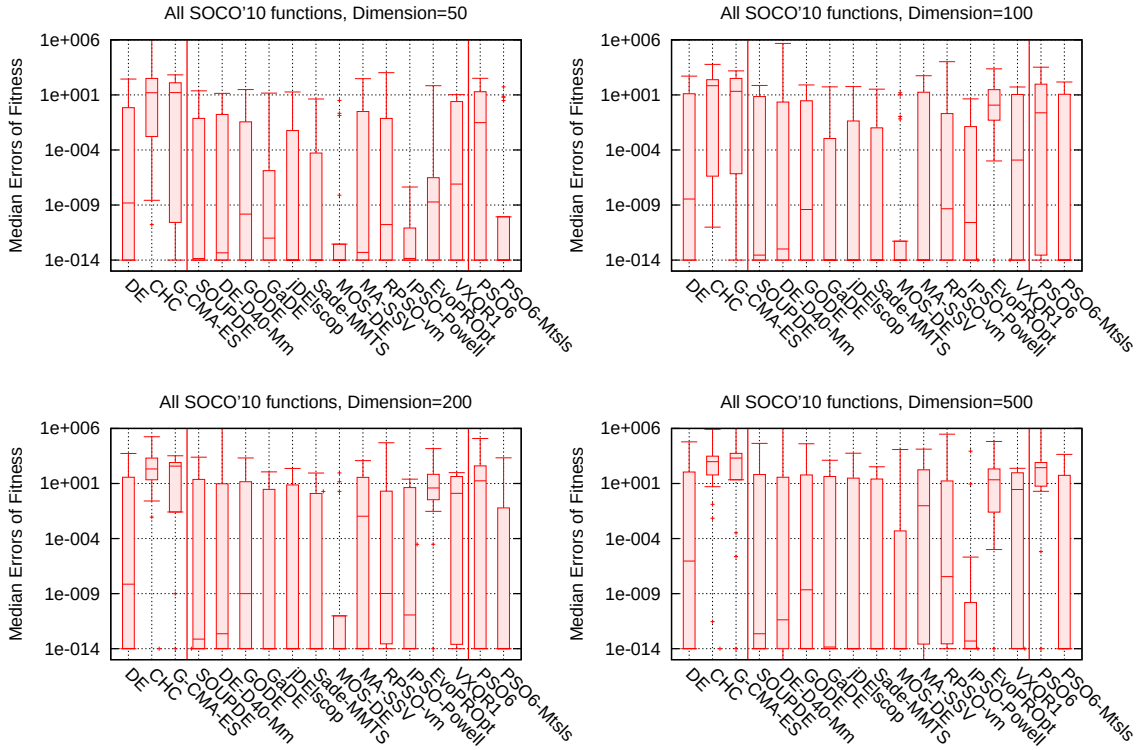


Figure 7.11: SOCO'10 function's fitness distributions of all featured algorithms and PSO6 with and without Mtsls, for dimensions 50, 100, 200, and 500

In general, our PSO6-Mtsls is statistically better than PSO6, and shows more accurate distributions than RPSO-vm and IPSO-Powell, the two other PSO versions evaluated in SOCO'10. An exception can be found for dimension 500 where IPSO-Powell also shows an accurate distribution, although with worse median value than our proposal. Concerning the remaining algorithms, PSO6-Mtsls significantly outperforms G-CMA-ES, CHC, DE, VXQR1, EvoPROpt, and MA-SSW. In 200 and 500 dimensions, our proposal also outperforms GODE and DE-D40-Mm. Nevertheless, the best ranked distributions correspond to the ones of MOS-DE, which is established as control method in Table 7.8. In spite of this, we can effectively check that adjusted p-values of PSO6-Mtsls are always higher than 0.05 (confidence level), which leads us to ensure that no statistical differences can be found between our proposal and MOS-DE. The remaining techniques featured in SOCO'10: IPSO-Powell, Sade-MMTS, jDElscop, GaDE, and SOUPDE are also in the group of similar algorithms (without statistical differences in their performance) with regards to our proposal and the control algorithm (MOS-DE).

From the point of view of the problem scalability, we can observe in Figure 7.11 that the performance of PSO6-Mtsls keeps competitive results even in dimension 500 with regards to the group of best compared algorithms. In fact, we have to notice that the median errors of PSO6-Mtsls are below the 0-threshold at least for 12 functions out of 19 (SOCO'10). Therefore, as shown in boxplots (Figure 7.11), our proposal resulted with a global median of 1.00E-14 for all dimensions.

Table 7.8: Average Friedman's rankings with Holm's correction ($\alpha = 0.05$) for SOCO'10 functions and featured algorithms

Algorithm	50		100		200		500	
	Rank	$Holm's_p$	Rank	$Holm's_p$	Rank	$Holm's_p$	Rank	$Holm's_p$
PSO6-Mtsls	5.894	0.16E+01	7.929	1.24E-01	6.952	2.40E-1	7.236	2.14E-01
PSO6	10.263	0.13E-02	11.763	7.12E-05	12.315	4.61E-06	12.947	8.40E-08
SOUPDE	7.578	0.62E+00	6.763	5.40E-01	7.553	1.44E-01	7.631	1.41E-01
DE-D40-Mm	9.342	5.53E-01	8.710	5.85E-02	8.578	3.90E-02	8.657	3.45E-02
GODE	8.657	0.15E+00	7.894	1.44E-01	7.973	1.02E-01	7.921	1.02E-01
GaDE	6.868	0.11E+01	5.631	1.26E+00	5.631	6.23E-01	5.552	6.38E-01
jDElscop	5.868	0.16E+01	5.315	1.26E+00	5.315	6.23E-01	5.052	6.38E-01
Sade-MMTS	6.263	0.16E+01	5.631	1.26E+00	6.157	5.47E-01	6.368	5.40E-01
MOS-DE	**4.921**	-	**4.315**	-	**3.973**	-	**3.921**	-
MA-SSV	11.736	4.45E-04	10.421	2.33E-03	9.131	1.64E-02	11.578	3.54E-05
RPSO-vm	9.552	0.52E-02	9.631	1.17E-02	9.210	1.29E-01	8.236	6.74E-02
IPSO-Powell	6.078	0.16E+01	8.210	1.22E-01	7.736	1.53E-02	6.173	6.30E-01
EvoPROpt	10.184	1.43E-01	12.421	1.05E-05	13.026	4.60E-07	12.526	1.95E-05
VXQR1	10.447	0.96E-03	9.868	7.71E-03	10.815	3.55E-04	10.000	2.27E-03
DE	10.026	0.18E-02	9.210	2.53E-02	9.131	1.64E-02	8.815	2.81E-02
CHC	16.157	1.11E-11	15.736	5.03E-11	16.052	2.67E-12	16.105	1.64E-12
G-CMA-ES	13.157	7.45E-06	13.368	4.93E-07	13.342	1.61E-07	13.473	7.73E-08

In this way, we can claim that our approach is also competitive as the problem dimensionality increases.

An interesting observation in this comparison concerns the performance of G-CMA-ES, which is relatively limited on SOCO'10 functions. Although this algorithm shows accurate results on non-separable functions *soco*3, *soco*5, and *soco*8, it has a moderate performance on separable unimodal and multimodal ones, such as *soco*4, *soco*6, *soco*7, as well as on non-separable hybrid composed (from *soco*12 to *soco*19). Taking into account that G-CMA-ES obtained the best results in the special session of CEC'05, we suspect that the existence (or not) of rotated functions, on which this algorithm shows highly accurate results, could influence its global performance with regards to other compared algorithms. We have to notice that a number of rotated functions (21 out of 25) are included in CEC'05, whereas practically none of them can be found in SOCO'10. A similar observation was made in [LMdOA+11], where the authors argued that the global performance of a given algorithm can be biased to certain function feature more expressed in the tackled benchmark. This motivated us to use an extended benchmark composed of 40 problem functions from CEC'05 and SOCO'10 (previously described in Table 3.2 in Chapter 3) to compare our proposal with G-CMA-ES, as well as with other related swarm intelligent approaches with local search methods. The results of this comparison are analyzed in the following section.

Comparisons on an Extended Benchmark

Table 7.9 contains the median of distribution errors obtained by G-CMA-ES [AH05], IPSO-Powell [MdOAS11], IPSO-Mtsls [MdOSVdED11], IACOr-Mtsls [LMdOA+11], and PSO6-Mtsls, out of 30 independent runs on the extended benchmark of 40 functions taken from SOCO'10 and CEC'05, for dimension 50. IPSO-Powell and IPSO-Mtsls are PSO versions that perform an incremental social learning mechanism for swarm size adaptation on continuous optimization func-

Table 7.9: Median Errors obtained by G-CMA-ES, IPSO-Powell, IPSO-Mtsls, IACOr-Mtsls, and PSO6-Mtsls, for dimension 50

F/A	G-CMA-ES	IPSO-Powell	IPSO-Mtsls	IACOr-Mtsls	PSO6-Mtsls
fsoco1	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
fsoco2	2.64E-11	**1.42E-14**	4.12E-13	4.41E-13	2.96E-12
fsoco3	**0.00E+00**	**0.00E+00**	6.38E+00	4.83E+01	8.47E-11
fsoco4	1.08E+02	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
fsoco5	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
fsoco6	2.11E+01	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
fsoco7	7.67E-11	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
fsoco8	**0.00E+00**	1.75E-09	2.80E-10	2.66E-05	**0.00E+00**
fsoco9	1.61E+01	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
fsoco10	6.71E+00	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
fsoco11	2.83E+01	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
fsoco12	1.87E+02	1.02E-12	**0.00E+00**	**0.00E+00**	**0.00E+00**
fsoco13	1.97E+02	**2.00E-10**	5.39E-01	6.79E-01	3.09E+00
fsoco14	1.05E+02	1.77E-12	**0.00E+00**	**0.00E+00**	5.37E+01
fsoco15	8.12E-04	1.07E-11	**0.00E+00**	**0.00E+00**	**0.00E+00**
fsoco16	4.22E+02	3.08E-12	**0.00E+00**	**0.00E+00**	**0.00E+00**
fsoco17	6.71E+02	**4.35E-08**	1.47E+01	6.50E+00	6.68E+00
fsoco18	1.27E+02	8.06E-12	**0.00E+00**	**0.00E+00**	6.21E+00
fsoco19	4.03E+00	1.83E-12	**0.00E+00**	**0.00E+00**	**0.00E+00**
fcec3	**0.00E+00**	8.72E+03	1.59E+04	8.40E+05	1.73E+02
fcec4	4.27E+05	2.45E+02	3.88E+03	**5.93E+01**	2.08E+02
fcec5	5.70E-01	4.87E-07	**7.28E-11**	9.44E+00	2.98E+03
fcec7	3.85E-14	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
fcec8	**2.00E+01**	**2.00E+01**	**2.00E+01**	**2.00E+01**	**2.00E+01**
fcec10	**9.97E-01**	8.96E+02	8.92E+02	2.69E+02	1.84E+02
fcec11	**1.21E+00**	6.90E+01	6.64E+01	5.97E+01	3.38E+01
fcec12	2.36E+03	5.19E+04	3.68E+04	1.37E+04	**6.97E+02**
fcec13	4.71E+00	3.02E+00	3.24E+00	**2.14E+00**	6.70E+00
fcec14	2.30E+01	2.35E+01	2.36E+01	2.33E+01	**2.27E+01**
fcec15	2.00E+02	2.00E+02	2.00E+02	**0.00E+00**	3.06E+02
fcec16	**2.15E+01**	4.97E+02	4.10E+02	3.00E+02	1.95E+02
fcec17	**1.61E+02**	4.54E+02	4.11E+02	4.37E+02	2.20E+02
fcec18	9.13E+02	1.22E+03	1.21E+03	9.84E+02	**8.25E+02**
fcec19	9.12E+02	1.23E+03	1.19E+03	9.93E+02	**8.25E+02**
fcec20	9.12E+02	1.22E+03	1.19E+03	9.93E+02	**8.25E+02**
fcec21	1.00E+03	1.19E+03	1.03E+03	**5.00E+02**	7.18E+02
fcec22	8.03E+02	1.43E+03	1.45E+03	1.13E+03	**5.00E+02**
fcec23	1.01E+03	**5.39E+02**	**5.39E+02**	**5.39E+02**	7.24E+02
fcec24	9.86E+02	1.31E+03	1.30E+03	1.11E+03	**2.17E+02**
fcec25	**2.15E+02**	1.50E+03	1.59E+03	9.38E+02	**2.15E+02**
#bests	11/40	14/40	18/40	21/40	**23/40**

tions. These two IPSO algorithms are hybridized with Powell's direction set [Pow64] and Mtsls (LS1) [TC08] local search procedures, respectively. IACOr-Mtsls consists of an Ant Colony Optimization algorithm also performing an incremental social learning mechanism as in the previous

Table 7.10: Average Friedman's rankings with Holm's correction ($\alpha = 0.05$) for SOCO'10 and CEC'05 functions

Algorithm	Rank	$Holm's_p$
PSO6-Mtsls	**2.48**	-
IACOr-Mtsls	2.68	5.71E-01
IPSO-Mtsls	3.08	1.79E-01
IPSO-Powell	3.35	4.41E-02
G-CMA-ES	3.38	4.36E-02

PSO versions, and also hybridized with Mtsls (LS1). In this way, we can compare our proposal with modern swarm intelligent approaches hybridized with the same and different local search methods. G-CMA-ES is a covariance matrix adaptation evolution strategy that performs frequent restarts with increasing population size. As commented before, we also compare our PSO6-Mtsls with G-CMA-ES on CEC'05 non-separable/rotated functions, where this last algorithm shows an impressive performance. In this table, the best median values for each function are represented in boldface, and the last row contains the global count of the number of best medians for each compared algorithm.

A first observation in Table 7.9 is that PSO6-Mtsls obtains the highest number of best median errors (23 out of 40), followed by IACOr-Mtsls, IPSO-Mtsls, IPSO-Powell, and finally G-CMA-ES. The statistical tests associated to these results are presented in Table 7.10, where we can effectively validate that our proposal is the best ranked algorithm, then working in this case as control method for the post-hoc Holm's correction. According to this, we can even ensure that IPSO-Powell and G-CMA-ES are statistically outperformed by our PSO6-Mtsls.

In this sense, a second observation is that our approach does not show statistical differences with regards to the other two swarm intelligent methods hybridized with Mtsls, that is, IACOr-Mtsls and IPSO-Mtsls. This led us to suspect that Mtsls is largely responsible for the accurate performance of these three approaches, in comparison with IPSO-Powell and G-CMA-ES. However, the difference in their ranking values seems to be mostly due to the contribution of their base methods: PSO6, IPSO, and IACO.

A final interesting observation concerns the different function features that our proposal can successfully tackle with regards to the four compared techniques. Table 7.11 shows a detailed comparison presented in form of (win, draw, lose) according to different features of the extended benchmark of 40 functions. In comparison with G-CMA-ES, our approach obtained a higher number of "wins" (better medians) on non-separable, multimodal, and rotated functions (as well as in non-separable and non-rotated). We have to notice that rotated functions correspond to CEC'05 benchmark on which G-CMA-ES was the best algorithm. In fact, our PSO6-Mtsls obtained 8 "wins" in CEC'05 and 10 in SOCO'10, in contrast with 5 and 1 "loses" in these two benchmarks with regards to G-CMA-ES. If we have a look on non-separable functions, our proposal obtains a higher number of "wins" than G-CMA-ES and IPSO-Powell, although the number of "draws" is higher in comparison with IACOr-Mtsls and IPSO-Mtsls. Once again, the effect that Mtsls induces on non-separable functions leads hybridized algorithms with this local search method to outperform other compared techniques. A similar behavior can be observed concerning multimodal functions with a high number of "draws" when comparing hybridized algorithms with Mtsls. Nevertheless, it is on rotated functions where PSO6-Mtsls shows a higher number of "wins" in comparison with all other algorithms. In this case, the base method PSO6 is responsible of the accurate performance, since it never obtained "loses" on rotated functions, excepting 5 in comparison with G-CMA-ES.

Table 7.11: Number of best median errors with regards to different functions features when comparing PSO6-Mtsls versus G-CMA-ES, IPSO-Powell, IPSO-Mtsls, and IACOr-Mtsls. The results are presented in form of (win, draw, lose)

Function's features	PSO6-Mtsls versus			
	G-CMA-ES	IPSO-Powell	IPSO-Mtsls	ACOr-Mtsls
Separable	(6, 1, 0)	(0, 4, 0)	(0, 3, 0)	(0, 4, 0)
Non sep.	(12, 4, 6)	(13, 9, 2)	(9, 12, 3)	(9, 12, 5)
Unimodal	(1, 2, 1)	(1, 5, 1)	(1, 5, 1)	(1, 5, 1)
Multimodal	(17, 3, 5)	(12, 8, 1)	(8, 10, 2)	(8, 11, 4)
Rotated	(7, 2, 5)	(7, 3, 0)	(7, 3, 0)	(7, 4, 0)
Non rot.	(10, 3, 1)	(6, 10, 2)	(2, 12, 3)	(2, 12, 5)
SOCO'10	(10, 3, 1)	(5, 10, 2)	(1, 12, 3)	(1, 12, 2)
CEC'05	(8, 2, 5)	(8, 3, 0)	(8, 3, 1)	(8, 4, 3)
Total	(18, 5, 6)	(13, 13, 2)	(9, 15, 3)	(9, 16, 5)

In summary, the local search method Mtsls (LS1) seems to be responsible of the successful performance of our proposal on non-separable and multimodal functions, whereas the learning procedure of PSO6 takes mostly part in rotated ones. PSO6-Mtsls shows more "wins" than "loses" in all comparisons, although the number of "draws" is higher when it is compared with IACOr-Mtsls and IPSO-Mtsls, e. g., the other swarm intelligent approaches hybridized with Mtsls.

7.7 Conclusions

In this chapter, we generalize and analyze the number of informants that take part in the calculation of new particles. For this, we have created a new version of Informed PSO, called PSOk with the possibility of managing any neighborhood size k, from 1 informant to all of them in the swarm (FIPS-ALL). The new proposal has been thoroughly analyzed from the point of view of the evolvability, and it has been also hybridized with a modern local search method (MTS) to tackle with non-separable problems more efficiently. A series of experiments and comparisons have been carried out in the scope of CEC'05 and SOCO'10 benchmarks of functions. The influence of the number of informants, the problem dimension, and the swarm size have been also analyzed.

In general, the following conclusions can be extracted after our work on this initial hypothesis:

1. A number of 6 informants in the neighborhood makes the algorithm to perform with high success in practically all tackled functions. This means that, at least for the popular continuous benchmarks, researchers should consider PSO6 instead of the standard PSO.

2. Using few informants (<4) leads the PSOk to show a positive fitness-distance correlation, although it find solutions with poor fitness values and far from global optima. With more than 10 informants, solutions are again correlated, although concentrating on small non interesting regions of the landscape. Using 6 informants is the best trade-off between fitness-distance and fitness quality.

3. In general, the higher the number of informants (involved in the velocity calculation), the longest the running time required.

4. Each PSOk version shows quite similar behavior in our experiments independently of the swarm size, and independently of the problem dimension. This means that PSOk is hav-

ing additional features making it scalable and resistent to constrained execution (memory-restricted, at least).

5. In the scope of the extended benchmark with 40 functions used here, we can ensure that our new variant, PSO6-Mtsls, statistically outperforms IPSO-Powell and G-CMA-ES, and is better ranked than IACOr-Mtsls and IPSO-Mtsls. The local search method Mtsls (LS1) seems to be responsible of the successful performance on non-separable and multimodal functions, whereas the learning procedure of PSO6 takes mostly part in rotated ones.

In general, we can state that PSO is a first class optimizer able of the best performance in present benchmarking for the continuous optimization.

As future work, we are interested in investigating other elemental features and learning procedures of the PSO algorithm, as well as to study other complementary methods to construct satisfactory hybrid approaches, capable of solving highly complex functions. Besides, we plan to perform analytical investigations on new benchmarks (BBOB, CEC'13, etc.) with different function characteristics and dimensions.

Part III

Real World Applications

Chapter 8

Gene Selection in DNA Microarrays

8.1 Introduction

In this chapter, we initiate a new part of this PhD Thesis in which our goal is to move apart from academic benchmarking and solve actual real world complex problems. We plan to test PSO on hard tasks with a clear real utility and real data. Despite this being an important goal, we must notice that going real world is usually requiring that much effort that the contents of these chapters are more balanced to the problem than to the algorithms. Our final aim is to step forward final users and applications to avoid missing this perspective as it can be seen in many present research works: we think this is an error and we want to learn from facing the problems themselves.

The problem addressed in this chapter is in the domain of Biology and Bioinformatics. The importance of the research in Health is that vast in the world of today that we think it is worth dedicating part of our efforts to at least one important problem. The selected one has been Feature Selection in DNA Microarrays, an activity open to research and at the same time used in modern hospitals for customized medicine.

DNA Microarrays ([PSS+94]) allow scientists to simultaneously analyze thousands of genes, thus providing important insights about cells' functions, since changes in the physiology of an organism are generally associated with changes in large gene ensembles of expression patterns. The vast amount of data that is involved in a typical Microarray experiment usually requires from scientists to perform a complex statistical analysis, with the goal of classifying the dataset into correct classes. The key issue in this classification is to identify significant and representative gene subsets that may be later used to predict class membership for new external samples. Furthermore, these subsets should be as small as possible in order to develop fast and low consuming processes for future class prediction. The main difficulty in Microarray classification versus other domains is the availability of a relatively small number of samples in comparison with the number of genes in each sample. In addition, expression data are highly redundant and noisy, and most genes are believed to be uninformative with respect to studied classes, as only a fraction of genes may present distinct profiles for different classes of samples.

In this context, machine learning techniques have been applied to handle large and heterogeneous datasets, since they are able to isolate the useful information by rejecting redundancies [RSA10, CW07]. Concretely, feature selection (gene selection in Biology) is often considered

as a necessary preprocess step in analyzing large datasets, as this method can reduce the dimensionality of the datasets and often results in better analyses [GWBV02, VLPD12, VH11].

Feature selection for gene expression analysis in cancer prediction often uses wrapper classification methods [KJ98] to determine a type of tumor, to reduce the number of genes to investigate in the case of a new patient, and also to assist in drug discovery and early diagnosis. Several classification algorithms could be used for wrapper methods, such as K-Nearest Neighbor (K-NN) [FH51] or Support Vector Machines (SVM) [CV95]. By defining clusters, a big reduction of the number of considered genes and an improvement of the classification accuracy can be finally achieved.

Definition 8.1.1. Let $F = \{f_1, ..., f_i, ..., f_n\}$ be a set of features; find a subset $F' \subseteq F$ that maximizes a scoring function $\Theta : \Gamma \rightarrow G$ such that $F' = argmax_{G \subset \Gamma}\{\Theta(G)\}$; where Γ is the space of all possible feature subsets of F and G a subset of Γ.■

A formal definition of the feature selection problem is expressed in Definition 8.1.1. Optimal feature selection is a complex problem proved to be NP-hard [NF77]. Therefore, we need efficient automated approaches to tackle it. Metaheuristics algorithms have been shown to be adequate tools for this matter, since they are capable of solving the feature selection accurately and efficiently for the large dimensions needed in Biology. Evolutionary Algorithms (EAs) and, specifically, Genetic Algorithms (GAs) have been successfully used in the past to tackle the gene selection of Microarrays ([AGNJT07, HDH07, HDH06, JKCN05]). All these approaches consist in using single population sequential algorithms which can achieve competitive performances (from the point of view of the quality of solution), but without considering other important aspects such as the computational effort and the time consumption.

Parallel metaheuristics have always been very popular in the literature [Alb05] and thus, there exists a large number of implementations and algorithms. Concretely, population based algorithms are naturally prone to parallelism, since most of their variation operators can be easily undertaken in parallel. Besides the the numerical benefits [Alb05, AD08] spatial distribution of individuals could induce to the learning procedure, the execution of many computational steps per time unit offers an additional profit in reducing the computing time of such numerically enhanced structured metaheuristics. Unfortunately, not much work has been done so far on parallel structured metaheuristics for feature selection [ZJP06], and no related approaches (to the best of our knowledge) have been developed for gene selection of Microarray datasets.

In this chapter, a structured particle swarm optimization is used for gene selection of high dimensional Microarrays datasets. The proposed algorithm, called Parallel Multi-Swarm Optimizer (PMSO), consists in running a set of parallel subPSOs algorithms forming an island model, where a migration operation exchanges solutions between those islands with a certain frequency. A feature selection mechanism is embedded in each subPSO for finding small samples of informative genes amongst thousands of them. The reported solutions, are then evaluated by means of their classification accuracy by using a SVM classifier, performing 10-fold cross-validation and final testing with external test datasets. As optimizer algorithm, we have decided to use the Geometric PSO [MCP07] (GPSO) (described in Subsection 3.1.3 of Chapter 3) for binary search spaces. There exists several binary PSO versions [KS98, Cle05, PFE05], although all of them consists of *ad hoc* adaptations from the original one, and their performances are usually improvable. As evaluated in [AGNJT07, GNA12a], the *three-parent mask-based crossover* (3PMBCX) used in GPSO makes the offspring inherit the shared selected features present in the three parents involved in the mating, hence making this operator especially suitable for the feature selection.

The contributions are noticeable, since the parallelization GPSO will be shown to improve existing algorithms in terms of computational effort and classification accuracy. Besides, the gene

ensembles found by this technique are successfully interpreted in the light of independent existing results from Biology to show their actual impact. The effectiveness of our proposal is analyzed on four well-known public datasets: Leukemia [GST+99], Colon [ABN+99], Lymphoma [Ali00] and Lung [GJH+02], discovering new and biologically challenging gene subsets, and identifying specific genes that our work suggests as significant ones. Comparisons with several recent state of art methods will show the effectiveness of our results in terms of computational time/effort, reduction percentage and classification rate.

We have organized this chapter as follows. In Section 8.2, we provide the reader with basic concepts about Microarrays technology. Section 8.3 gives the details of our Parallel Multi-Swarm Optimizer algorithm for feature selection. Experimental results and comparisons are presented in Section 8.4, including performance analyses and biological validation of the obtained gene subsets. Conclusions and further work are finally given in Section 8.5.

8.2 DNA Microarrays

A DNA Microarray consists of an arrayed series of thousands of DNA molecules spotted in different positions in a matrix structure [PSS+94]. These DNA molecules, that correspond to particular genes, are mixed with cellular cDNA molecules (called labeled or colored DNA) during a hybridization process. A cDNA molecule is obtained from cellular RNA or mRNA during a labeling process showing the relative expression level of each molecule. RNA molecules are isolated from a particular cell type or tissue comprising of a complex mixture of different RNA transcripts. The abundances of individual transcripts in the mixture reflect the different expression levels of the corresponding genes. This process is called hybridization, after which abundant sequences will generate strong signals, while rare sequences will generate weak signals.

Microarrays are normally used to compare gene expression levels within a sample or look at differences in the expression of specific genes across different samples, such as a few samples of one disease or healthy and unhealthy tissues. A gene expressed only in the disease sample, for example, might represent a useful drug target. This is especially appropriate in cancer analysis, since it allows us to discriminate against tumoral tissues and normal ones. Several gene expression profiles obtained from tumors such as Leukemia, Colon, and Lung cancers have been published.

Algorithm 10 Pseudocode of PMSO

```
1: do in parallel for each i ∈ {1, ..., m}
2: S_i ← initializeSwarm(Ss_i)
3: while not stop_condition do
4:    iterate S_i for n steps  /* GPSO evolution */
5:    for each S_j ∈ τ(S_i) do
6:       send ρ particles in φ_s(S_i) to S_j
7:    end for
8:    for each S_j such that S_i ∈ τ(S_j) do
9:       receive ρ particles from S_j
10:      replace ρ particles in S_i according to φ_r
11:   end for
12: end while
```

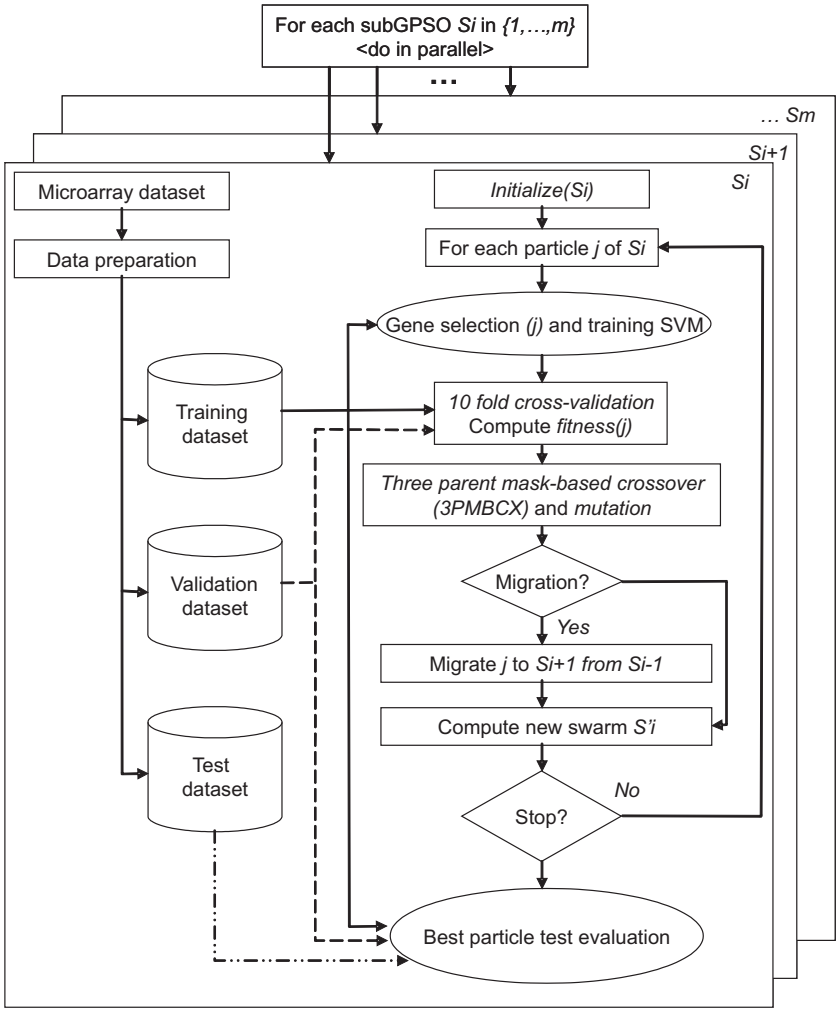


Figure 8.1: General model of PMSO for gene selection and classification of Microarrays. Training, validation, and testing mechanisms are embedded into the parallel algorithm

8.3 PMSO for Gene Selection

In analogy with Parallel Genetic Algorithms (PGAs) [AL05, SA12], we define our Parallel Multi-Swarm GPSO (PMSO) as a pair $\langle \mathcal{S}, \mathcal{M} \rangle$, where $\mathcal{S} = \{S_1, ..., S_m\}$ is a collection of m swarms (populations) and $\mathcal{M}$ is the migration policy. The main parameters of the migration policy constitute a five-tuple $\mathcal{M} = \langle \sigma, \rho, \phi_s, \phi_r, \tau \rangle$, where $\sigma \in \mathbb{N}$ (*migration gap*) denotes the number of iterations in every subswarm between two successive exchanges of particles (steps of isolated evolution), $\rho \in \mathbb{N}$ (*migration rate*) is the number of particle copies that undergo migration in each exchange; ϕ_s and ϕ_r are two functions which respectively decide how to select emigrant particles and what particles have to be replaced by incoming immigrants. The topology is denoted by $\tau : \mathcal{S} \rightarrow 2^{\mathcal{S}}$, e. g., a unidirectional ring, in our case. Algorithm 10 shows the pseudocode of PMSO.

Two alternatives exist for ϕ_s: *random* and *best*. The first one refers to randomly selecting any particle to be migrated; the second one selects the best-known particle in the island subswarm. Concerning ϕ_r, also two strategies are considered *always* and *if_better*. The former refers to always replacing the worst particle(s) by the incoming emigrant(s); the later replaces the worst particle(s) only if it is worse than the incoming emigrant(s).

Finally, concerning the topology (τ), a unidirectional *ring* is considered where each subswarm sends (and receives) particles to (from) the two consecutive nearest subswarms in the ring. It must be noted that the subswarms proceed asynchronously, giving rise to loosely coupled contiguous epochs of computation and then communication. For this work, we have selected the asynchronous version since it usually provides a better performance than the synchronous one [AT01].

Gene Selection and Classification Scheme

Following the basic scheme of solution encoding used in feature selection, PMSO provides a binary encoded particle (vector) where each bit represents a gene in the dataset. If a bit is 1, it means that this gene is kept in the reduced subset, while 0 indicates that the gene is not included. Therefore, the particle length is equal to the number of genes in the initial Microarray dataset.

As illustrated in Figure 8.1, where a general model of our PMSO is provided, the particles of each subswarm (S_i), representing gene subsets, are evaluated by means of a SVM classifier and 10-fold cross-validation as follows: each gene subset (codified by a particle) is divided into ten subsets, nine of them constituting the training set and the remaining one used as the validation set. The SVM is trained using the training set and then the accuracy obtained (number of correct classifications by the SVM once trained) is evaluated on the validation set [CV95]. This evaluation is repeated ten times, each one alternating the used validation set. This method reinforces the validation process, so that the final accuracy value is the resulting average of the ten validation folds. Such a strong validation is necessary when the number of samples is low regarding the number of features, which is the case for this work. As a final evaluation, the resulting subset solution is evaluated on the external test set, thus obtaining the final accuracy (standard protocol recommended in supervised learning).

Fitness Function

A fitness function is needed to guide the search by assigning to any tentative solution a quality value. Once the accuracy value and the number of genes are known, the fitness function that we propose is calculated according to the following:

$$f(x) = (100 - acc) + \lambda \cdot \frac{\#(genes\ in\ subset)}{\#(total\ genes)}, \tag{8.1}$$

$$being,\ \lambda = 10^{\lfloor log(\#(total\ genes))+1 \rfloor} \tag{8.2}$$

The objective here consists of maximizing the accuracy and minimizing the number of genes. For convenience (only minimization of fitness), the first factor is presented as $(100 - acc)$ and the second one is normalized in order to control the trade off between these two factors. A constant value λ (which depends on the total number of genes) is used in this normalization. Therefore, if the number of features in the subset is high (with regards to the total number of genes in the original dataset), then the fitness function promotes the reduction of features. Otherwise, if the number of features in the subset is small, then this fitness function promotes the improvement in accuracy.

Table 8.1: Usage details of the four Microarray Datasets

Dataset	#genes	Classes	#Train	#Test	#Total
Colon	2,000	Cancer	20	20	40
		Normal	11	11	22
Lymphoma	4,026	Ac B-like	17	6	23
		Ce B-like	19	5	24
Leukemia	7,129	AML	11	14	25
		ALL	27	20	47
Lung	12,533	MPM	16	15	31
		ADCA	16	134	150

8.4 Experimental Results

In this section, the experiments are described by first discussing the Microarrays datasets used, then the experimentation setup, the analysis of results and comparisons, and a global discussion. We have implemented the proposed PMSO algorithm for gene selection in C++ following the *skeleton* architecture of the MALLBA library [ALGN+07]. For the SVM classifier we have used a set of classes provided by the LIBSVM [CL02] library for training, validation, and testing. These classes were coupled with those of the PMSO for this evaluation phase.

8.4.1 Microarray Datasets and Data Preprocessing

The instances used are classified into four well-known datasets taken from real-word Microarray experiments. All of them were taken from the public repository of Kent Ridge Bio-medical Dataset (`http://datam.i2r.a-star.edu.sg/datasets/krbd/index.html`). In particular:

- The ALL-AML Leukemia dataset consists of 72 Microarrays experiments with 7,129 gene expression levels. Two classes exist: *Acute Myeloid Leukemia* (AML) and A*cute Lymphoblastic Leukemia* (ALL). The complete dataset contains 25 AML and 47 ALL samples. The original dataset is divided into a training set of 38 samples and a test set of 34 samples.

- The Colon Tumor dataset consists of 62 Microarray experiments collected from colon-cancer patients with 2,000 gene expression levels. Among them, 40 tumor biopsies are from *tumors* and 22 (*normal*) are from healthy parts of the colon of the same patient.

- *Types of Diffuse Large B-cell Lymphoma* dataset consists of 47 tissue samples, 24 of them are from *germinal centre B-like group* while the rest 23 are *activated B-like group*. Each sample is described by 4,026 genes.

- The Lung Cancer dataset involves 181 experiments with 12,533 gene expression levels. Classification occurs between *Malignant Pleural Meso-thelioma* (MPM) and *Adenocarcinoma* (ADCA) of the lung. In tissue samples there are 31 MPM and 150 ADCA.

Table 8.1 summarizes the original organization of training and testing samples of the four used Microarrays. The test sets of Leukemia and Lung were taken from the original repositories provided by the authors. In Colon and Lymphoma, only training sets are available in the original repositories, and for this reason, new test and training sets have been here generated for these two datasets by randomly (uniformly) extracting samples from the original one as stated in Table 8.1. These datasets were selected because of their different dimensions and gene organizations, constituting a heterogeneous test-bed to better support our conclusions.

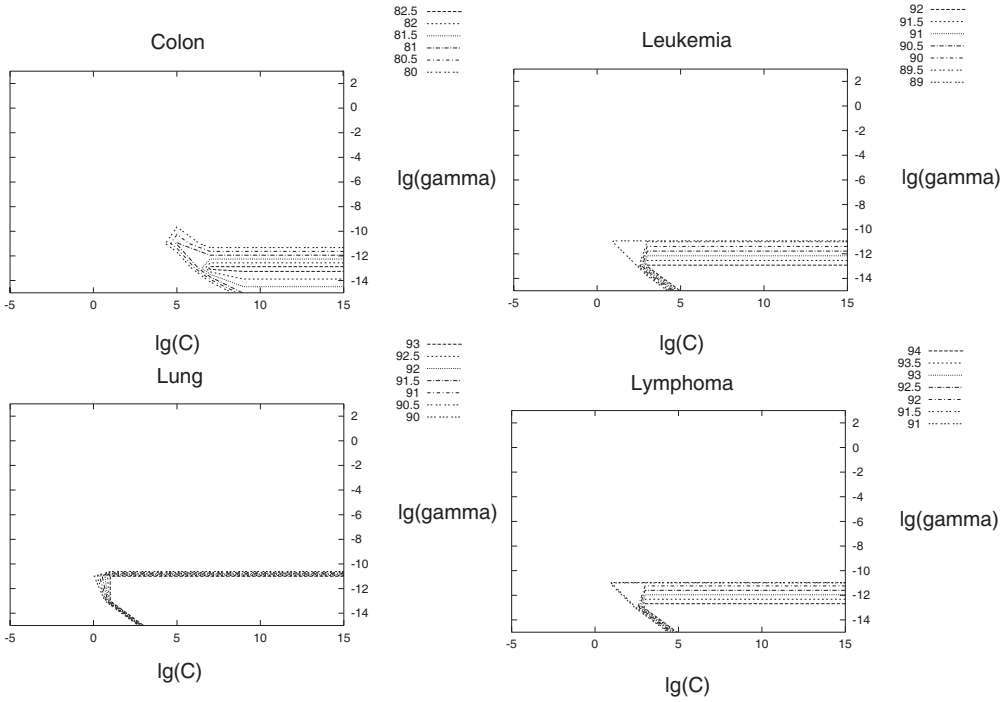


Figure 8.2: Plots of the different grid-search parameters evaluated in SVM for each dataset

Expression levels of training and test sets were normalized separately in order to scale their intensities, thus enabling a fair comparison between the different datasets. Therefore, for each attribute a_j (gene) with $j \in \{1 \cdots \#attributes\}$, and for each sample x_k (expression) with $k \in \{1 \cdots \#samples\}$, a scaling operation to $[-1, 1]$ was performed resulting $a'_j(x_k)$ by using the following equation (as LIBSVM recommends):

$$a'_j(x_k) = 2 \cdot \frac{a_j(x_k) - min_j}{max_j - min_j} - 1, \tag{8.3}$$

where max_j and min_j correspond to the maximum and minimum gene expression values for attribute a_j. Reductions of genes by removing them according to thresholds were not made previously, and so we make the task of correct classification harder by leaving even clearly non-functional genes for the algorithm to remove. Uninformative genes to the classifier could nevertheless be informative ones to the metaheuristic algorithm, since bad solutions are quite important to avoid guiding the search towards low quality regions of the search space.

8.4.2 Experimental Settings

All experiments were carried out using a cluster of PCs with Linux O.S. (Suse 9.0 with kernel 2.4.19) and a Pentium IV 2.8GHz processor, with 1GB of RAM. The PMSO algorithm was independently executed 30 times on the Microarray datasets in order to have statistically meaningful conclusions. Each one of these PMSO executions performed 500 iterations.

In the parameters setup, an optimal configuration of the SVM classifier is crucial, since it influences the training effectiveness. Therefore, the main Kernel (RBF) parameters γ and C coefficient [CV95], were systematically optimized (as recommended in LIBSVM [CL02]) in a preprocess phase using grid-search with cross-validation. Basically, this consists in identifying the best combination of both parameters in a rank of bounded values (for example, $C = 2^{-5}, 2^{-5}, \cdots 2^{-15}, \gamma = 2^{-15}, 2^{-13}, \cdots, 2^{3}$). Figure 8.2 plots the traces of the different grid-search parameters for each training dataset. The resulting parameters are :

- Leukemia: $C = 8$ and $\gamma = 0.0001220703125$ (accuracy = 92.0%)
- Colon: $C = 128$ and $\gamma = 0.0001220703125$ (accuracy = 82.5%)
- Lymphoma: $C = 8$ and $\gamma = 0.000030517557$ (accuracy = 94.0%)
- Lung: $C = 2$ and $\gamma = 0.0001220703125$ (accuracy = 93.0%)

These parameters were set using the SVM classifier independently of the PMSO, in order to obtain an accuracy as higher as possible (as shown in parenthesis). The parameters of PMSO were set using the previously tuned SVM classifier. Finally, after the PSMO executions, a testing process was made on each resulted subset. In this process, the parameters of SVM were tuned using the grid-search method for the resulting subsets and test sets separately.

Both sets of PMSO parameters, the ones defining the distributed model and those specific to GPSO, were set according to a preliminary study. As a result of this study, our PMSO has been run with the best configuration using 160 particles organized in 1, 2, 4 and 8 subswarms (with 160, 80, 40 and 20 particles each subswarm/processor), thus constituting four different configurations regarding the number of subswarms. In these configurations, we have considered $\sigma = 100$ and $\rho = 1$. The selection/replacement strategies choose respectively the *best* particle to be sent and replaces *if_better*. The migration topology is a *unidirectional ring*. Finally, concerning the GPSO parameters, we have used similar present, historic, and social weights $w_1 = w_2 = w_3 = 0.33$. The probability of performing mutation is 0.01.

8.4.3 Performance Analysis

Table 8.2 shows the results obtained by PMSO, out of 30 independent runs, using the four different configurations (number of swarms in column 2). As a robustness indicator, column 3 denotes the number of executions in which the amount of genes (#Genes) in the resultant subset is lower than 5 (very good result from a biological point of view). Following the standard methodology when comparing classification rates, the average and standard deviation of the obtained accuracies (Accuracy 1) and the number of genes are reported in columns 4 and 5, respectively. Column 6 shows the reduction percentage[1] of each computed subset regarding the original datasets. In the last column, the accuracy percentage (Accuracy 2) of the tuned stand-alone SVM on each complete dataset (before reduction) is presented.

Several observations can be made from Table 8.2. First of all, the accuracy rate and the number of genes obtained by PMSO with 8 swarms (8-Swarm PMSO) is the best in all the cases for each dataset. This confirms that our parallel approach is clearly the way to go if a high accuracy and computational effort are needed, which is the case in actual labs. Statistical differences were found in these results regarding Colon, Lymphoma and Lung datasets. These differences in distributions

[1] $Reduction = 100 - \frac{\#(genes\ in\ subset)}{\#(total\ genes\ in\ dataset)}$

Table 8.2: Results of PMSO using 1, 2, 4, and 8 subswarms' configurations. Columns denote the number of subswarms, the hit rate (expressed as subsets with less than 5 genes), the accuracy rate, the number of genes, and the reduction percentage

Datasets	Swarms	(#Genes< 5)	Accuracy 1 (%)	#Genes	Reduction (%)	Accuracy 2 (%)
Colon	1	30	79.98±5.61	2.40±0.71	88	82.5
	2	30	82.04±5.24	2.23±0.49	89	
	4	30	85.53±3.61	2.06±0.24	90	
	8	30	**85.55±4.06**	**2.00±0.00**	**90**	
Lymphoma	1	22	96.94±3.32	4.06±1.15	90	92.0
	2	24	96.75±2.87	3.60±1.08	92	
	4	28	97.12±2.72	3.23±0.76	92	
	8	30	**97.87±2.56**	**2.86±0.49**	**93**	
Leukemia	1	30	98.00±2.21	3.00±0.74	96	94.0
	2	24	98.00±2.60	4.00±0.92	95	
	4	28	98.00±2.54	3.00±0.88	96	
	8	28	**98.00±1.85**	**3.00±0.77**	**96**	
Lung	1	30	96.00±3.69	3.00±0.63	98	93.0
	2	30	97.39±3.13	2.63±0.70	98	
	4	30	97.00±3.17	2.66±0.64	98	
	8	30	**97.39±1.99**	**2.26 ± 0.44**	**99**	

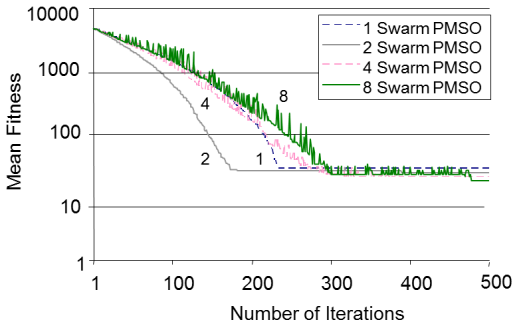


Figure 8.3: Mean performance of PMSO in four different swarm distributions: 1, 2, 4, and 8 subswarms. The mean fitness (in logarithmic scale) is plotted versus the number of iterations

(out of 30 independent runs) were statistically assessed by using the parametric/non-parametric procedure as explained in Section 2.1.3 of Chapter 2. A confidence level of 95% ($\alpha = 0.05$) was always applied in order to check whether statistically differences can be found, or not.

Secondly, in comparison with the accuracy percentage (Accuracy 2) of the SVM on each complete dataset, our results show better accuracy percentages in 14 out of 16 configurations. This is a true improvement since the reduction of features in all the cases with regards to the original datasets is around 90% in our model while SVM is using all genes. Specifically, our PMSO with 4 and 8 swarms always obtains better accuracies (with subsets lower than 5 genes) than the stand-alone SVM on the complete datasets.

When analyzing the internal behavior of these configurations of PMSO, we can clearly see that the ones distributed with more subswarms (4 and 8) show a smoother convergence than the others (i.e., diversity is better preserved). We suspect that the migration mechanism in PMSO introduces also a high diversity which helps the initial exploration in all configurations. This behavior is

Table 8.3: Mean time of execution in microseconds (ms) performed by our 8-Swarm PMSO running on 1, 2, 4 and 8 processors

Dataset	$\overline{T_1}$	$\overline{T_2}$	$\overline{T_4}$	$\overline{T_8}$
Colon	2.69E+09	1.42E+09	7.60E+08	5.28E+08
Leukemia	5.14E+09	2.83E+09	1.52E+09	1.05E+09
Lymphoma	3.05E+09	1.69E+09	8.80E+08	4.74E+08
Lung	7.64E+09	4.19E+09	2.38E+09	1.39E+09

clearly observable in Figure 8.3 where the mean performances through the evolution steps of 30 independent runs of each PMSO are plotted. In this figure, the lines represent the mean of the fitness obtained by the subswarms at each iteration. For this reason, the lines corresponding to PMSO with 4 and 8 subswarms show peaks that represent migrations injecting diversity in the receptor swarm, hence altering the averages of the global evolution fitness. This beneficial behavior is only slightly observed in PMSO with 2 subswarms, and non-existent in PMSO with 1 subswarm.

8.4.4 Speedup Analysis

One of the most important parameters for measuring the efficiency of a parallel algorithm is the *Speedup* (Sp). The standard formula of the speedup is represented in Equation 8.4, where m is the number or processors used, $\overline{T_1}$ is the mean time of execution of all the subswarms of the algorithm in 1 processor, and $\overline{T_m}$ is the mean time of execution of the swarms in parallel on m processors.

$$Sp = \frac{\overline{T_1}}{\overline{T_m}} \tag{8.4}$$

Table 8.3 shows the mean time of execution in microseconds (ms) performed by our 8-Swarm PMSO running on 1, 2, 4 and 8 processors. The most time consuming execution corresponds to $\overline{T_1}$ in Lung dataset which takes about 2.12 hours (7.64E+09 ms). This time is reduced down to 23.16 minutes (1.39E+09 ms) when using 8 processors ($\overline{T_8}$) with the same dataset. Specifically, the 8-Swarm PMSO running on 8 processors obtains a reduction in the computational time of 69.8% when dealing with Lung. This is a clear improvement concerning the time consumption since Lung is the larger dataset we have tackled.

More precisely, in order to work with a measure proportional to the number of processors employed we calculated the speedup using the execution times obtained. For this, we followed the standard methodology described in [Alb02]. Figure 8.4 shows a graphical representation of the speedup performed by our 8-Swarm PMSO algorithm executed in 1, 2, 4 and 8 parallel processors. In this graphic, the linear speedup is represented by a dotted line. That is, a linear (ideal) speedup is obtained when $Sp = m$, and hence, in the execution of an algorithm with linear speedup, doubling the number of processors ideally means doubling the speed. The remaining lines represent the speedup of the 8-Swarm PMSO on Leukemia, Colon, Lymphoma, and Lung datasets.

As we can observe in Figure 8.4, all the speedup values are close to the linear one, being all of them higher than 5 when running in 8 processors. This way, the mean efficiency[2] reported is $E = 70\%$ for all datasets which is a very good value for facing still larger problems in the future. The best efficiency is obtained when running in 4 processors being $E = 85\%$. In short, our PMSO provides an efficient outcome even in the presence of such a high dimensionality of the solutions (from 2,000 to 12,533 genes), stating a low overhead of communications and thus being a globally scalable technique.

[2]Mean efficiency $E = \frac{Sp}{m} \times 100(\%)$

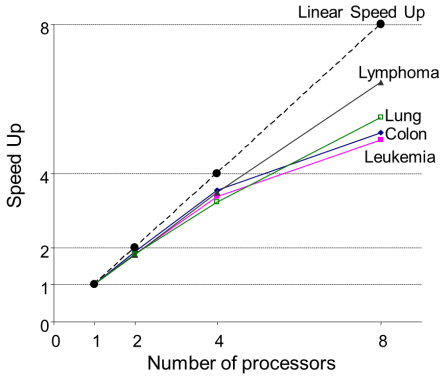


Figure 8.4: Linear (ideal) speedup versus actual speedup of the 8-Swarm PMSO executed in 1, 2, 4 and 8 processors

Table 8.4: Comparison of PMSO versus PGA, both configured with 8 islands

Dataset	Algorithm	(#Genes≤5)	Accuracy (%)	#Genes
Colon	PMSO	30	**85.55±4.06**	**2.00±0.00**
	PGA	30	81.45±4.22	2.10±0.30
Lymphoma	PMSO	30	**97.87±2.56**	**2.86±0.49**
	PGA	30	94.44±3.72	3.60±1.28
Leuklemia	PMSO	28	98.00±1.85	**3.00±0.77**
	PGA	30	98.55±1.55	**3.15±0.85**
Lung	PMSO	30	**97.39±1.99**	2.26±0.44
	PGA	30	93.43±4.86	**2.00±0.00**

8.4.5 PMSO versus Parallel Island Genetic Algorithm

In this section, we compare the results obtained by our PMSO operating with 8 subswarms (8-Swarm PMSO), against a Parallel Island Genetic Algorithm (PGA) [Alb05] also operating with 8 subpopulations. This way, we look to make a comparison not only versus the standard SVM but also versus other parallel techniques popular in the literature.

The PGA has been configured as follows: the whole population consists of 160 individuals (codifying gene subsets) organized in 8 subpopulations with 20 individuals each one. We have considered $\sigma = 100$ and $\rho = 1$ (explained in Section 8.4.2). The selection/replacement strategies choose, respectively, the *best* individual to be sent and replaces *if_better*. The migration topology is also a *unidirectional ring*. Finally, concerning the genetic algorithm parameters we have used a generational strategy of reproduction (e.g. number of offspring equal to number of parents), two-point crossover (probability of crossover 0.9), and simple bit-flip mutation (probability of mutation 0.01) as applied to GPSO in Section 8.3. The evaluation task in PGA has followed the same procedure as for PMSO. The evaluation task in PGA has followed the previously explained parameter setting of SVM for classification, 10-fold cross-validation, and fitness function. A final testing of the resultant subsets has been also accomplished.

The PGA was implemented in C++ using the MALLBA library. For the experiments, we have used the same pool of machines as explained in Section 8.4.2. The PGA was executed 30 times, each one of these executions performing 500 iterations.

Table 8.4 shows the results obtained by PGA together with those obtained by PMSO for each dataset. Column 3 denotes the number of executions in which the amount of genes in the resulted

Table 8.5: Comparison with six other works. Columns indicate the average of the accuracy and the number of genes in the final subsets. The most accurate results are in boldface. Cells with unavailable values are marked with "-"

Dataset	Author	Accuracy (#G≤5)	Accuracy (5<#G≤10)	Accuracy (#G>10)
Colon	[HDH07]	-	91.20(8)	-
	[HDH06]	-	**99.41(10)**	-
	[JKCN05]	-	-	94.12(37)
	[LCJM04]	**93.55**(4)	-	-
	8-S PMSO	87.61(**3**)	88.70(10)	**94.22(20)**
Lymphoma	[HDH07]	93.31(5)	-	-
	[LI02]	-	-	90.00(13)
	8-S PMSO	**97.87(3)**	**96.48(10)**	**98.70(20)**
Leukemia	[HDH07]	91.50(3)	-	-
	[HDH06]	-	-	**100**(25)
	[LCJM04]	87.55(4)	-	-
	8-S PMSO	**98.00(3)**	**96.31(10)**	98.15(**20**)
Lung	[AGNJT07]	99.00(4)	-	-
	[LCJM04]	-	98.34(**6**)	-
	8-S PMSO	**99.38(2)**	**99.47**(10)	**100(20)**

subset is smaller than 5. The columns 4 and 5 indicate the average and standard deviation of the accuracy rate and the number of genes in the final subsets, respectively. In bold we mark the most accurate results. As we can observe, PMSO obtains higher percentages of accuracy than PGA in Colon, Lymphoma and Lung datasets. These differences in accuracy have been statistically assessed by using the same procedure explained in Section 8.4.3. For Leukemia dataset, the differences in the results of PGA and PMSO are statistically negligible. Concerning the number of genes in the resulting subsets, PMSO obtains the smaller subsets in Colon, Lymphoma, and Leukemia. Only in Lung dataset, PMSO obtained slightly larger subsets than PGA which always obtained subsets with 2 genes. In conclusion, we can state that PMSO performs better than PGA in the scope of the problem datasets studied here.

8.4.6 PMSO versus Other Approaches

In our aim of providing a thorough assessment of our results, in this section we additionally compare the results obtained by our PMSO operating with 8 subswarms (8-S PMSO) with six other approaches found in the literature. We have to notice that this study is very heterogeneous because other authors do not offer all the needed information and the algorithms are quite varied; hence, an exhaustive comparison can not be made. However, a simple comparison with other reported results is still useful.

The values are reported in Table 8.5, in terms of the average of the accuracy and the number of genes in the final subsets. In order to provide a more understandable comparison, these results are separated according to intervals of different numbers of genes (3 last columns). The accuracy reported by the 8-Swarm PMSO correspond to subsets achieved with 20, 10, and ≤5 genes (#G). We saved the subsets of genes generated with the specified lengths in each independent execution. As we can see, for subsets with more than 10 genes, the accuracy rate reported by our 8-Swarm PMSO is the best in 3 out of 4 datasets beating all the existing algorithms. In them, the only worse accuracy value was found for the Leukemia dataset [HDH06] but our algorithm is finding solutions with a lower number of genes. With the smaller subsets (≤5 genes), our approach reports the highest accuracy in Lymphoma, Leukemia and Lung. For Colon, Liu et al. [LCJM04] reported slightly higher accuracies than 8-Swarm PMSO, although with a larger number of genes. Finally, with subsets of between 5 and 10 genes, our approach is the best in Lung, but with lower accuracy

Table 8.6: Top 11 genes ranked by Golub et al. which were also obtained with PMSO on the Leukemia dataset

Rank	Index	Accession	Gene Description
1	4847	X95735_at	Zyxin
5	1834	M23197_at	CD33 antigen (differentiation antigen)
6	2020	M55150_at	FAH Fumarylacetoacetate
8	3320	U50136_rna1_at	Leukotriene C4 synthase (LTC4S) gene
15	4499	X70297_at	CHRNA7 Cholinergic receptor, nicotinic, alpha polypeptide 7
14	2267	M81933_at	CDC25A Cell division cycle 25A
16	5039	Y12670_at	LEPR Leptin receptor
18	6376	M83652_s_at	PFC Properdin P factor, complement
20	6041	L09209_s_at	APLP2 Amyloid beta (A4) precursor-like protein 2
24	2354	M92287_at	CCND3 Cyclin D3
28	461	D49950_at	Liver mRNA for interferon-gamma inducing factor(IGIF)

than in Huerta et al. [HDH06] and in Hernandez et al. [HDH07] in Colon. Therefore, we can claim that PMSO shows a competitive performance in comparison with state of the art algorithms.

8.4.7 Biological Analysis of the Results

In this section we provide a biological analysis of the computed gene subsets. Similar biological studies have been carried out in relevant papers in the past [WZ07, DRIE$^+$08]. Then we show the broad impact of using PMSO, able to compute biological ensembles of genes that have been independently suggested in the domain (e.g. [GST$^+$99] and [Ali00]).

In Figure 8.5, a graphical distribution of the most frequently obtained genes in 30 independent executions of the 8-Swarm PMSO are reported. We have used the Leukemia dataset, since it is the one commonly studied by other related works in the literature. From this distribution, a brief selection of the 11 most overlapped genes (the ones in bold with frequency ≥ 3 in Figure 8.5) out of all the computed subsets, are described in Table 8.6. We can remark that all of these genes were also reported in the list of the 30 most important genes (selected from 7,129 ones in Leukemia) suggested in Golub et al. [GST$^+$99]. Hence, we arrange the genes by means of the rank assigned in the Golub et al. [GST$^+$99] list (column 1 in the referenced table). This way, the gene U50136_rna1_at that we obtained with frequency 7 was ranked in [GST$^+$99] in eighth position. Moreover, the first ranked gene (X95735_at Zyxin) in [GST$^+$99], that we obtained with frequency 5, is the only gene that is capable of discerning between AML and ALL samples in just one split.

The genes reported in Figure 8.5 (in boldface) were also selected as the most informative genes in recent specialized works. Concretely, in [DRIE$^+$08] a Monte Carlo method was used for feature selection and supervised classification on Leukemia and Lymphoma datasets. In [WZ07] a Nearest Shrunken Centroid (NSC) was proposed to classify the Leukemia dataset. Both works considered the genes X95735_at and M23197_at as the most important ones, which matches our main results.

Concerning the Lymphoma dataset, three of the most frequently selected genes by 8-Swarm PMSO are: G1622X, G1618X and G2399X. These genes were also reported in the list of the 30 most important genes (selected from 4,026 ones in Lymphoma) suggested in Alizadeh et al. [Ali00]. Moreover, genes selected from Leukemia and from Lymphoma in this work have been validated by means of the GO system[3], finding in all of them associations from Human proteins. Therefore, the selection of validated genes, also discovered in specialized publications in this area, leads us to claim the great ability of our PMSO for selecting informative genes in actual DNA Microarrays.

[3]the Gene Ontology `http://www.geneontology.org/`

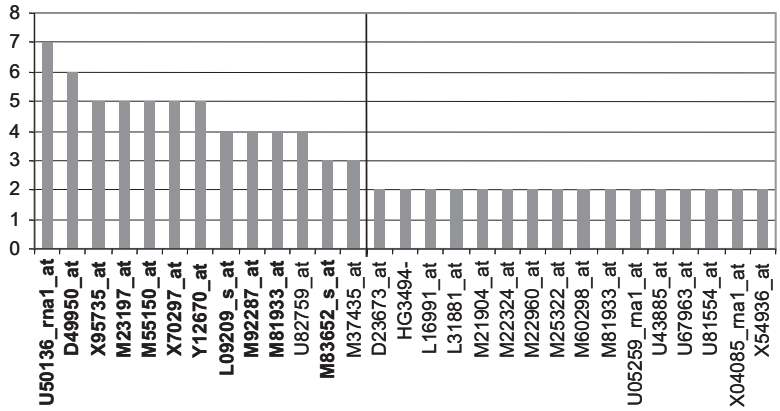


Figure 8.5: Distribution of the most frequently genes obtained (in 30 independent runs) by our 8-S PMSO on the Leukemia dataset

8.5 Conclusions

In this chapter, a Parallel Multi-Swarm Optimizer (PMSO) algorithm is proposed for the first time for gene selection of high dimensional Microarray datasets. PMSO has proven to be both efficient and accurate. It is able to improve sequential algorithms, in terms of computational effort (Efficiency of 85%). It has been experimentally assessed with different population structures on four well-known cancer datasets, identifying specific genes that our work suggests as significant ones. Concretely, with regard to the Leukemia and Lymphoma datasets, we could confirm that the most frequently PMSO reported genes are also the most relevant genes suggested in the original publications (Golub et al. and Alizadeh et al., respectively) concerning these Microarrays. Comparisons with several recent state of the art methods also show competitive results close to 100% classification rate and very few genes per subset (4, 5 genes) obtained in 458 out of 480 (30×16) independent executions.

As for future work, we are interested in developing and testing several combinations of other metaheuristics with classification methods in order to discover still unseen and better subsets of genes using specific Microarray datasets. In this sense, the use of ensemble classifiers could contribute notably to the DNA Microarrays analysis.

Chapter 9

Optimizing Software Communication Protocols

9.1 Introduction

In this chapter we now go to a different field of real world applications, that of allowing new services in smart cities by endowing cars with communication abilities to connect each other. The very first problem in such a big endeavour (this will change our daily experience) is to make the communication network exists. And this results only possible when the software protocols allowing data exchange are optimized, since they standard configurations are unable to allow smooth maintained communications between moving cars in a real city.

Vehicular Ad Hoc Networks (VANETs) [HFB07] are fluctuating networks composed of a set of communicating vehicles (nodes) equipped with devices which are able to spontaneously interconnect to each other without any pre-existing infrastructure. This means that no service provider is present in such kind of networks, as it is usual in traditional or in mobile cellular communication networks. The most popular wireless networking technology available nowadays for establishing VANETs is the IEEE 802.11b WLAN, also known as WiFi (*Wireless Fidelity*). New standards such as the IEEE 802.11p and *WiFi Direct* are promising but still no available to perform real tests with them. This implies that vehicles communicate within a limited range while moving, thus exhibiting a topology that may change quickly and in unpredictable ways. In such kind of networks, previous to its deployment, it is crucial to provide the user with an optimal configuration of the communication protocols in order to increase the effective data packet exchange, as well as to reduce the transmission time and the network use (with their implications on higher bandwidth and lower energy consumption). This is specially true in certain VANET scenarios (as shown in Figure 9.1) in which buildings and distances discontinue communication channels frequently, and where the available time for connecting to vehicles could be just one second.

The efficient protocol configuration for VANETs without using automatic intelligent design tools is practically impossible because of the enormous number of possibilities. It is especially difficult (e.g., for a network designer) when considering multiple design issues, such as highly dynamic topologies and reduced coverage. In addition, the use of exact techniques is also impracticable due to the time spent during the great number of simulations required. All this motivated us to use of metaheuristic algorithms, which arise as well-suited tools to solve this kind of problems.

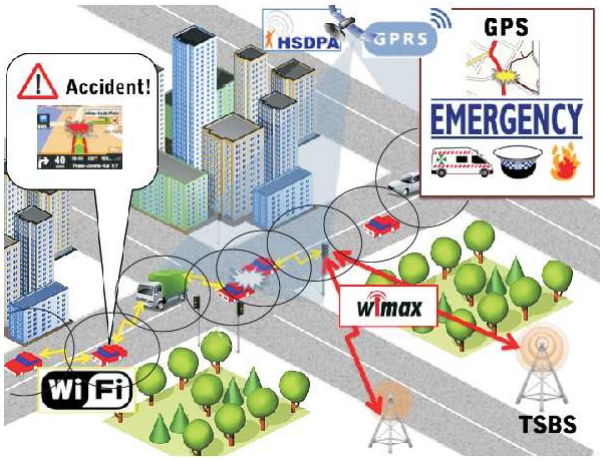


Figure 9.1: Typical urban VANET scenario. Circles represent the WiFi coverage of vehicles

In this chapter, we face the optimal File Transfer Protocol Configuration (FTPC) in VANETs by means of a Particle Swarm Optimization approach. This problem lies in the core of any VANET application, and thus optimal configuration is a major concern. We also use other four optimization algorithms, since this is a new field and their relative advantages are still unclear. Indeed, we can not find results for comparisons in the literature since only manual (human expert) VDTP configurations were made so far. These algorithms are other swarm intelligence technique: Differential Evolution (DE); two evolutionary algorithms: Genetic Algorithm (GA) and Evolutionary Strategy (ES); and a trajectory search technique, Simulated Annealing (SA). All these techniques have been previously described in Chapters 2 and 3 of this thesis. We have chosen these algorithms because they constitute a representative subset of well-known metaheuristics (population and trajectory based algorithms), with suitable operators for real parameter optimization, and with heterogeneous schemes of population and evolution. This way, we offer a set of initial results allowing future comparisons with other modern and traditional techniques.

For our tests, two typical car-to-car environment instances have been defined: urban and highway VANETs, both in special connection to the work done in the *CARLINK CELTIC European Project* [car] for linking cars[1]. We rely both on a flexible simulation structure using *ns-2* [ns2] (a well-known realistic VANET simulator), and real tests for optimizing: the *transmission time*, the *number of lost packets*, and the *amount of data transferred*. One additional contribution of this work is to provide the specialist with a useful platform, embedded within *ns-2*, to configure network protocols and hence obtaining a fair QoS control in VANETs.

The remaining of this chapter is organized as follows. In the next section, we briefly describe the most relevant related works found in the current literature. In Section 9.3, we introduce the Optimal File Transfer Configuration problem. In Section 9.4, the optimization strategy and fitness function are described. Experimental results and comparisons are presented in Section 9.5, including performance, scalability, and technical analyses of the resulted VANET protocol configurations. Conclusions and future work are drawn in Section 9.6.

[1]CARLINK CELTIC European Project, in URL http://carlink.lcc.uma.es

9.2 Literature Overview

Few related works can be found in the specialized literature concerning the use of metaheuristics for the optimization of Mobile Ad Hoc Networks (MANETs). Vanhatupa et al. [VHH06] proposed a flexible Genetic Algorithm for optimizing channel assignment in Mesh wireless networks. In that work, the network capacity was increased by 20% while keeping the coverage above 80%. In Alba et al. [ADL+07], a specialized cellular Multi-Objective Genetic Algorithm (cMOGA) was used for finding an optimal broadcasting strategy in urban MANETs, obtaining in this case three objectives fronts with coverage, bandwidth, and duration as performance metrics. The use of multi-objective techniques in this kind of works provides the specialists with a range of non dominated solutions which can help them in the decision making process. Nevertheless, the use of (mono-objective) aggregated functions allows us the possibility of weighting the objectives and assign more (or less) importance to them for better guiding the search. This way, in Dorronsoro et al. [DDBA08], six versions of GAs (panmictic and decentralized) were evaluated and successfully used in the design of ad hoc Injection Networks. From a different point of view, and due to its specific design, Ant Colony Optimization (ACO) has been successfully adapted for implementing new routing protocols for MANETs (Di Caro et al. [CDG05]), as well as for resource management (Chiang et al. [CCAB07]). Nevertheless, in these two last cases, the routing load provoked by the internal operations of ACOs makes these approaches unfeasible for large networks. More recently, Huang et al. [HCH09] proposed a new routing protocol based on a PSO to make scheduling decisions for reducing the packet loss rate in a theoretical VANET scenario.

In our work, besides of using the optimization technique itself as a protocol algorithm, our main contribution consists of improving the performance of an existing protocol by optimally tuning its parameters. In this way, we will hopefully obtain optimal configurations in the network design phase without incorporating extra management load to the actual network operation. Following this research line, we have to mention other recent studies performed in the scope of this thesis. In these studies, standard routing protocols [GNTA10] AODV [GNA10] and OLSR [TGNA12] have been optimally tuned with PSO, as well as other metaheuristics.

9.3 Problem Overview

The optimal File Transfer Configuration consists in optimizing the main parameters required by an application communication protocol. This protocol, called VDTP (Vehicular Data Transfer Protocol) [ATL06], operates on the transport layer protocols of VANETs, allowing the *end-to-end* file transfer. This implies that, considerations about the *multi-hop* interconnection mode and routing issues can be avoided, since they are carried out by the previous down layer protocols (e.g. UDP, DSR, IP, etc.). Therefore, the different vehicles that constitute the nodes in a given VANET can exchange complete files of information to each other by using VDTP.

9.3.1 Vehicular Data Transfer Protocol

VDTP is a connectionless protocol which operates on DSR [JMB01], a routing protocol for multi-hop wireless ad hoc networks. In VDTP, the communication process is carried out by both a file **petitioner**, which tries to download a file, and a file **owner**, which stores the file. This transfer protocol operates by using the following packets: *FIRQ (File Information Request)*, *FIRP (File Information Reply)*, *DRQ (Data Request)*, and *DRP (Data Reply)*. As shown in Figure 9.2 (a), once the file petitioner knows the name and the location of a given file, it starts the communication by

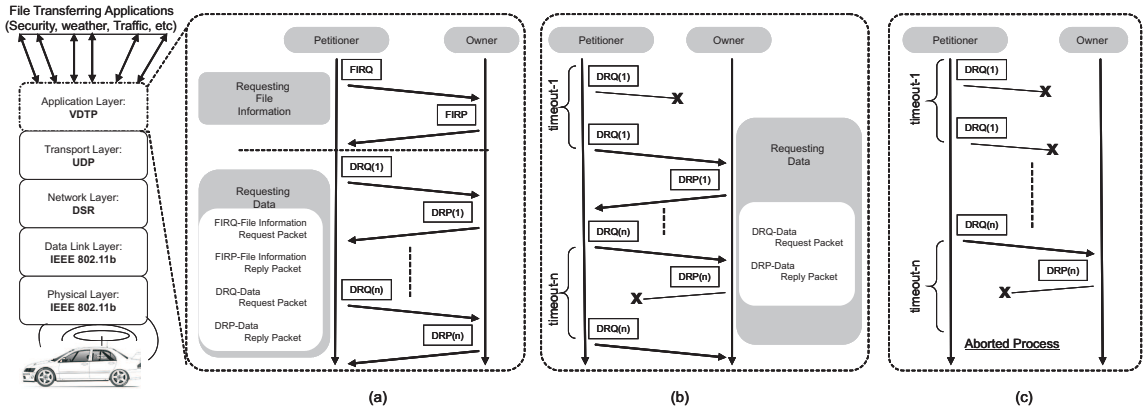


Figure 9.2: VDTP operation modes: (a) a complete file exchange is done; (b) timeout expiration and retransmission; (c) communication refused

using the FIRQ packet in order to obtain the file size. Then, the petitioner waits for this information which is sent by the owner by means of a FIRP packet. After receiving the information about the file size, the petitioner computes the number of segments in which the file will be split, dividing the file size by the **chunk_size**. The petitioner starts the transfer by sending a DRQ(1) packet asking for the first segment of the file; then it waits for the first data chunk sent by the owner which uses the DRP(1) packet. This operation is repeated by both, petitioner and owner, until transferring the last chunk DRP(n), and hence making up the complete file.

In VANETs, it is usual to work in a hostile medium which can provoke a high number of lost packets during the communication process. In this sense, VDTP provides the specialist with several mechanisms based on timers and counters, in order to solve such issues. The *timeout* mechanism controls the waiting time until a concrete DRQ or FIRQ packet has to be resent (**retransmission_time**). Figure 9.2 (b) shows an example of how the DRQ and the DRP packets are lost (and retransmitted) after an established timeout. The *counter* mechanism controls the number of DRQ/FIRQ packets that have been resent. As shown in Figure 9.2 (c), after a previously specified number of retransmissions (**total_attempts**) of the same DRQ/FIRQ packets, the communication between the vehicles is refused.

Since we are interested in finding the best possible configuration of VDTP, we have focused on the three aforementioned parameters: *chunk_size*, *retransmission_time* and number of *total_attempts*. Therefore, a given configuration (representing a solution of the problem) is a vector of three real values (*chunk_size*, *total_attempts* and *retransmission_time*). The range of each parameter is:

- *chunk_size*: $\mathbb{R}^+ \in [128 \cdots 524, 288]$ bytes (524,288bytes = 512 Kbytes)
- *total_attempts*: $\mathbb{R}^+ \in [1 \cdots 250]$ attempts
- *retransmission_time*: $\mathbb{R}^+ \in [1 \cdots 10]$ seconds

These ranges were stated following the CARLINK consortium requirements for VANETs applications [car].

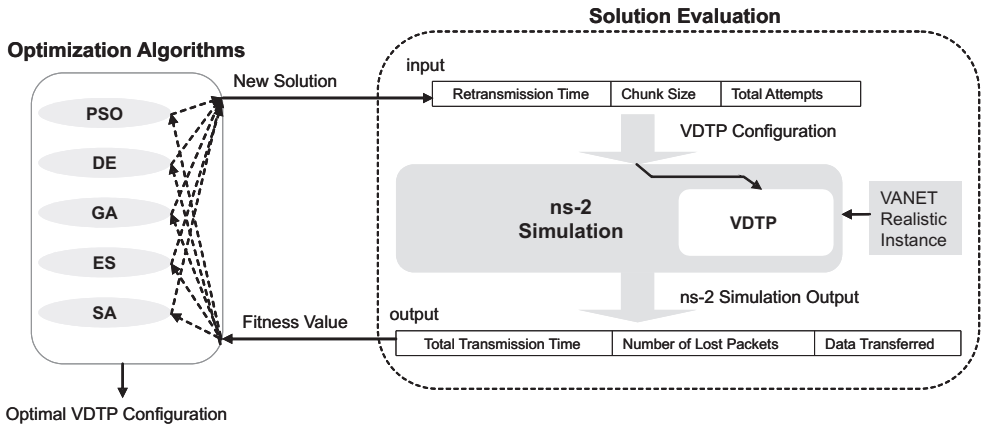


Figure 9.3: Optimization strategy for VDTP configuration in VANETs. The algorithms invoke the ns-2 simulator each solution evaluation

9.4 Optimization Strategy

Our optimization strategy for this problem is composed by basically two main parts: an optimization algorithm and a simulation procedure. The optimization part is carried out (independently) by our PSO (in this case Standard PSO 2007 [PCG11]), or by one of the selected metaheuristics. All of them are specially adapted to find optimal (or quasi-optimal) solutions in continuous search spaces (which is the case in this work). The simulation process is a way of assigning a quantitative quality value to the factors regulating VDTP, thus leading to optimal configurations of this protocol tailored to a given scenario. This procedure is carried out by means of the *ns-2* [ns2] simulator in which we have implemented the VDTP protocol for sending files in VANETs.

For each optimization algorithm, the evaluation of solutions is carried out by means of the simulation component. As Figure 9.3 illustrates, when a given algorithm generates a new solution it is immediately used for configuring the VDTP. This configuration evaluates the quality of the solution by using the received *retransmission time*, *chunk size*, and *total number of attempts*, as explained in Section 9.3.1. Then, *ns-2* is started and maps a given VANET scenario instance, taking its time in evaluating the scenario with buildings, signal loss, obstacles, vehicles, speed, covered area, etc., under the circumstances defined by the three control parameters optimized by the algorithm. After the simulation, *ns-2* returns the global information about the *transmission time* required for sending the file, the *number of lost packets* generated during the simulation, and the *amount of data* exchanged between vehicles. This information is used to compute the *fitness* function.

Fitness Function. Since *ns-2* operates by simulating (and averaging) many potential variations scenario all fitting the actual vehicle system, there is a possibility of obtaining different fitness values even using the same VDTP configuration (solution s). Therefore, in order to provide each solution with a fitness value as reliable as possible, a single evaluation of one solution requires $N = 10$ internal simulations, computing the global fitness ($fitness(s)$) as the mean of all *ns-2* results (Equation 9.1).

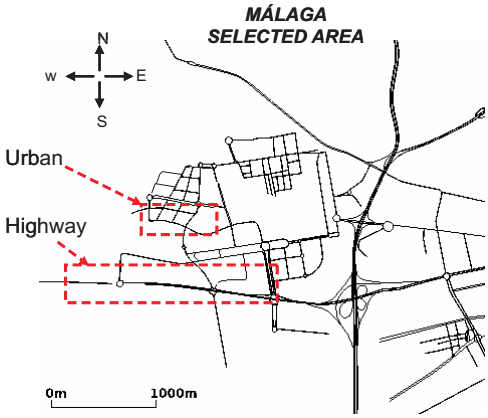


Figure 9.4: Selected area map of Malaga for our VANET instances. Urban and Highway areas are enclosed by dotted lines

$$fitness(s) = \frac{1}{N}\sum_{i=1}^{N}\frac{transmission\_time_i(s) + lost\_packets_i(s)}{\log(data\_transferred_i(s) + C)} \tag{9.1}$$

In this equation, $i \in [1 \cdots 10]$ is the number of simulations per solution evaluation. The factor $C = 2$ avoids division zero if there is no data transference, preventing a possible error in the fitness calculation. The data transferred is presented in logarithmic scale in order to make up for the difference in the range of values. This way, the algorithm looks for minimizing the global fitness[2].

9.5 Experiments

We have used the C++ implementation of Standard PSO 2007, as well as the other four algorithms provided by the MALLBA library [ALGN+07]. The simulation phase is carried out by running *ns-2* simulator v-2.31 [ns2]. For the experiments, we have made 30 independent runs of each algorithm on machines with Pentium IV 2.4 GHz core, 1 GB of RAM and O.S Linux Fedora core 6.

9.5.1 Instances: VANET Scenarios

We have created two simulation VANET scenarios (instances) from real urban and highway areas of Malaga, Spain (selected areas in Figure 9.4). These instances have been generated following the real tests carried out by experts in the scope of the CARLINK project, with the aim of obtaining as different as possible conditions of speed, number of vehicles, obstacles, signal noise, network use, etc. Therefore, we can analyze in both scenarios the behavior and performance of the compared algorithms, as well as the differences in the resulting VDTP configurations in terms of communication efficiency. Furthermore, we can compare these automatically generated configurations against the ones used in the real experiments by human experts in CARLINK [ATL08a, ATL08b].

[2] A multi-objective evaluation [Deb01] was not taken into account since objectives are not necessarily opposed themselves.

Table 9.1: VANET instances specification

Parameter	Value
Propagation model	Two Ray Ground
Carrier Frequency	2.472 GHz
Channel bandwidth	5.5 Mbps
Wifi Channel	13
Link Layer: transceiver	PROXIM ORiNOCO PCMCIA (IEEE 802.11b)
Link Layer: antenna gain	7 dBi (Omnidirectional)
Mac protocol	802.11-b
Routing Protocol	DSR
Transport Protocol	UDP
Application Protocol	VDTP
File transfers	20 sessions

- The *Urban* instance covers an area of 120,000 m^2 including buildings and semaphores. We have used *VanetMobiSim*[ATL07] for generating a realistic simulation mobility model where vehicles move randomly according to real traffic rules. A number of 30 vehicles move with a velocity between 30 km/h and 50 km/h, and 20 of them trying to send and receive a file of 1,024 kBytes.

- The *Highway* instance covers a stretch of 1 km with two directions without buildings and semaphores. In this case, the absence of obstacles is made up for the handicap of the high speed of vehicles, which also interferes the communication among vehicles. We have also used *VanetMobiSim*[ATL07] for generating a realistic simulation mobility model where vehicles move randomly according to real traffic rules. In the Highway VANET, a number of 30 vehicles move with a velocity between 80 km/h and 110 km/h, and 20 of them trying to send and receive a file of 1,024 kBytes size.

The resulted communication environments of Urban and Highway instances, including directions and mobile nodes (vehicles), were mapped in the *ns-2* simulator following the VANET specifications of devices and protocols[3] summarized in Table 9.1. The *ns-2* mobility trace definitions for both instances are also publicly available[4] .

9.5.2 Parameter Settings

In our experiments, all studied algorithms were configured in order to perform 1,000 solution evaluations per run. At each one of these solution evaluations, *ns-2* performs 10 independent simulations of the target scenario with the same protocol configuration as stated in Section 9.4. Therefore, all population based algorithms used here (PSO, DE, GA, and (μ, λ)-ES) were configured with 20 individuals, performing 50 generational steps.

Table 9.2 summarizes the remaining parameters specific to each algorithm. In this table, parameters corresponding to fitness values marked in bold were selected as the most accurate after a set of initial tuning experiments, for the two instances. In these, a number of 5 combinations of parameters per algorithm and VANET instance were tested performing 10 independent runs per combination, hence resulting a number of 500 additional executions.

[3]DSR (Dynamic Source Routing), UDP (User Datagram Protocol), and VDTP (Vehicular File Transfer Protocol).
[4]Mobility traces in URL `http://neo.lcc.uma.es/staff/jamal/portal/?q=content/malaga-scenario`.

Table 9.2: Different combinations and results of the preliminary parameter tuning

Algorithm	Parameter	Values				
	Instances	Results				
PSO	φ_1	2.0	2.0	2.0	2.0	2.0
	φ_2	2.0	2.0	2.0	2.0	2.0
	w	0.1	0.3	0.5	0.7	0.9
	Urban	1.952	1.978	**1.634**	2.766	3.280
	Highway	5.676	4.622	**4.1761**	5.283	6.045
DE	Cr	0.1	0.3	0.5	0.7	0.9
	μ	0.9	0.7	0.5	0.3	0.1
	Urban	4.027	2.647	2.241	1.866	**1.742**
	Highway	7.255	5.622	4.776	4.734	**4.663**
GA	P_{cros}	0.2	0.4	0.6	0.8	1.0
	P_{mut}	0.8	0.6	0.4	0.2	0.1
	Urban	2.701	2.245	1.953	**1.908**	2.077
	Highway	5.216	4.848	**4.380**	4.490	4.609
ES	P_{cros}	0.1	0.3	0.5	0.7	0.9
	P_{mut}	0.9	0.7	0.5	0.3	0.1
	Urban	4.920	3.878	3.031	2.606	**2.151**
	Highway	7.836	6.877	6.240	**5.783**	5.923
SA	T	0.2	0.4	0.6	0.8	1.0
	Urban	4.922	1.978	2.785	**1.634**	3.744
	Highway	7.665	5.201	4.820	**4.424**	4.683

Table 9.3: Final fitness values with regards to Urban and Highway VANET scenarios. Columns 3 contains the mean and standard deviation (Std. Dev.) of the fitness values in 30 independent runs. Columns 4, 5, and 6 show the minimum, median, and maximum values of fitness, respectively

Instance	Algorithm	Mean ± Std. Dev.	Minimum	Median	Maximum
Urban	PSO	**1.6346 ± 0.2899**	0.9077	**1.7809**	**1.8918**
	DE	1.7423 ± 0.3717	**0.7389**	1.8658	2.0228
	GA	1.9086 ± 0.2260	0.8799	1.9731	2.1614
	ES	2.1517 ± 0.1266	1.8862	2.1222	2.4246
	SA	2.7850 ± 0.8718	0.8730	2.1663	3.8025
Highway	PSO	**4.1761 ± 0.2556**	3.3301	4.2513	**4.4554**
	DE	4.6631 ± 0.9328	2.7145	4.2272	7.0531
	GA	4.3805 ± 0.8695	**2.5345**	4.1918	5.8608
	ES	5.7833 ± 0.9705	3.8836	6.1347	6.9421
	SA	4.4246 ± 0.7401	3.1498	**4.0855**	5.7922

9.5.3 Results and Comparisons

In this section we present the results obtained by the five studied algorithms when solving the optimal File Transfer Configuration (FTC) problem on VDTP. Table 9.3 shows the resulting fitness values regarding the Urban and Highway VANET scenarios in terms of the mean, the standard deviation, the minimum (best fitness), the median, and the maximum (worst fitness) found in 30 independent runs of every algorithm.

For the Urban scenario, we can observe in Table 9.3 that PSO obtained the best result in terms of the mean fitness. This best mean value leads us to believe that using the PSO the resulting VDTP ends in an efficient communication which is fast and accurate between vehicles. In addition, the best median and maximum values were also obtained by PSO, although the best minimum (e. g. the best VDTP configuration found for Urban) was reached by DE. This is an expected value, since DE generally shows a pronounced exploitative behavior (using a parametrization close to the standard one) [PSL05], while PSO tends to have an explorative performance using a high inertia (as in this study $w = 0.5$) [ES00]. Similar results can be observed for the Highway scenario, in which PSO obtained the best mean fitness value again. For this instance, PSO also showed the lowest value of standard deviation. This implies a considerable advantage, since it provides our

Table 9.4: Friedman's and Wilcoxon's Rankings ($\alpha = 0.05$), with PSO as control algorithm

Algorithm	Urban		Highway	
	Rank	Wilcoxon's$_p$	Rank	Wilcoxon's$_p$
PSO	**1.27**	-	2.17	-
DE	1.83	0.047	3.67	0.001
GA	3.07	0.001	1.97	0.453
ES	4.33	0.001	4.97	0.001
SA	4.50	0.001	**1.83**	0.371

model with a high robustness, which is a crucial issue when designing VANETs. In terms of the minimum fitness, GA and DE obtained the best VDTP configurations for the Highway scenario. The worst configuration was obtained by ES.

In order to provide such comparison with statistical meaningful, we have applied a Signed Rank test of Wilcoxon [Wil87] to the distributions of the aforementioned results. We have used this non-parametric[5] test with confidence level of 95% (*p-value*=0.05), which leads us to ensure that these results are statistically different if they result in *p-value*<0.05. Table 9.4 contains the resulted *p-value* of applying the Signed Rank test to PSO (the one with the best mean fitness) in comparison with the remaining of algorithms, hence confirming the differences in results. As we can observe in this table, PSO is statistically better than all compared algorithms for the Urban instance. Only DE shows a *p-value* (0.047) close to 0.05, being lower in any case. Concerning the Highway instance, PSO shows the best signed rank, although not far from GA and SA.

Additionally to Signed Rank test, a general comparison can be made using the Friedman [She07] statistical test by means of which the algorithms are sorted in a ranked list. Table 9.4 also shows the Friedman's ranking (columns 2 and 4) of the compared algorithms in Urban and Highway instances (the best ranked algorithm is in boldface). For Urban instance, PSO and DE are the best ranked algorithms, while SA is the worst performing technique. Nevertheless, for Highway scenario, SA obtains the best rank, whereas PSO is located in the third position.

Theses statistical results lead us to think that, in spite of the global best behavior of PSO, the different requirements implicit to both instances implies that each algorithm can show quite different results depending on the VANET scenario on which it operates. For example, DE shows a competitive performance in Urban scenario whereas it is the second worst in Highway. The opposite example can be observed in GA and SA which show weak results in Urban but highly competitive ones in Highway. Therefore, the VANET designer can select the optimization model more suited to his/her requirements, and choose the best option for each studied VANET scenario.

9.5.4 Performance Analysis

We present now a performance study which lying in analyzing the best fitness value resulted from each function evaluation, during the whole evolution process of a given algorithm. Figure 9.5 illustrates the graphs of the best fitness values (communication cost) obtained through the median execution in Urban and Highway instances.

We can observe in these two plots how PSO and DE tend to converge in the same range of solution evaluations, although they could improved their fitness even in the final steps of the evolution process. GA shows a similar trend as the former ones but it is subjected to an early stagnation. Finally, the different behaviors observed in ES, and specifically in SA, for Urban

[5]The distributions violate the condition of normality required to apply parametric tests (Z Kolmogorov-Smirnov = 0.009)

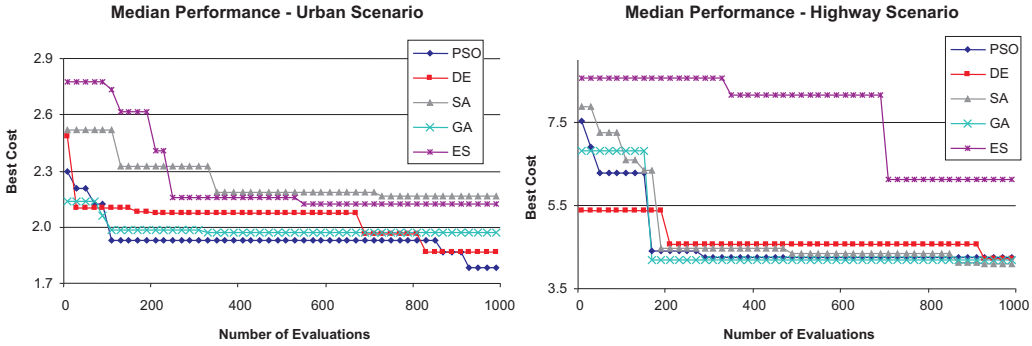


Figure 9.5: Median performance in Urban and Highway scenarios

Table 9.5: Mean execution time (seconds) per independent run of each algorithm for Urban and Highway scenarios

Instance	Algorithm	T_{best} (seconds)	T_{run} (seconds)
Urban	PSO	4.68E+03	7.95E+03
	DE	4.37E+03	7.12E+03
	GA	3.48E+03	6.68E+03
	ES	5.46E+03	9.00E+03
	SA	**2.18E+03**	**4.76E+03**
Highway	PSO	1.39E+03	2.19E+03
	DE	9.82E+02	2.10E+03
	GA	8.83E+02	1.56E+03
	ES	9.84E+02	1.47E+03
	SA	**5.85E+02**	**8.45E+02**

and Highway instances, confirm us the high dependency of these two algorithms to each different VANET instance, meaning that they are not robust enough for this application.

Concerning the mean run time that each algorithm has spent in the experiments, Table 9.5 shows both, the mean time in which the best solution was found T_{best} and the global mean run time T_{run}, for Urban and Highway scenarios. In general, SA shows the shortest times to find the best solution for the two VANET instances. We suspect that despite its temperature mechanism, SA quickly falls in local optima hence obtaining weak results in Urban scenario. Nevertheless, this behavior can be an advantage for Highway scenario where SA obtained accurate solutions with a fast performance. As expected in PSO and DE, they have spent closed executions times for the two VANET instances since they have similar internal operations. This resemblance in time consumption was also registered in the two evolutionary algorithms, GA and ES.

As a summary, the algorithms use between 9.00E+03 and 4.76E+03 seconds for the Urban scenario (150 and 80 minutes, respectively), and between 2.19E+03 and 8.45E+02 seconds for Highway scenario (60 and 23 minutes, respectively). This relative low overhead in the protocol design is completely justified by the subsequent benefits obtained in the global data transmission time and loss of packets once the VANET is physically deployed as observed in the following analysis.

9.5.5 Scalability Analysis

Once we have analyzed the performance of the five algorithms in two different VANET scenarios, we study in this section how do various network sizes affect the performance of these optimization

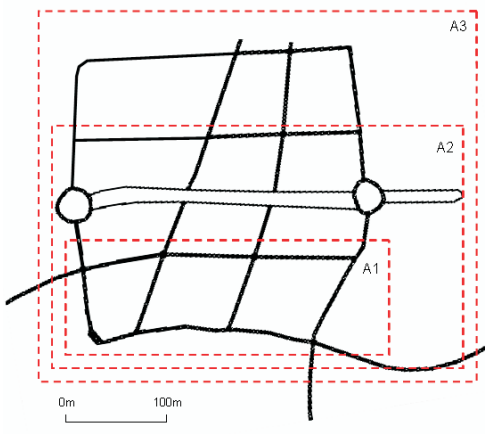


Figure 9.6: Three scaling urban areas from Malaga. Each area is a separate VANET instance

Table 9.6: Performance comparison in terms of mean fitness and mean optimization time (T_{best}) of the three scaled Urban VANETs

Algorithm	Mean Fitness			T_{best}		
	$Urban_{A1}$	$Urban_{A2}$	$Urban_{A3}$	$Urban_{A1}$	$Urban_{A2}$	$Urban_{A3}$
PSO	**1.6346±0.2899**	**1.3920±0.2831**	**3.6763±0.4435**	7.95E+03	5.93E+03	1.20E+04
DE	1.7423 ± 0.3717	1.4504 ± 0.1885	3.9186 ± 0.7419	7.12E+03	1.10E+04	1.43E+04
GA	1.9086 ± 0.2260	1.4100 ± 0.1235	3.6829 ± 0.5063	6.68E+03	9.81E+03	1.41E+04
ES	2.1517 ± 0.1266	1.5462 ± 0.6023	3.7799 ± 0.6227	9.00E+03	8.99E+03	1.50E+04
SA	2.7850 ± 0.8718	2.3880 ± 1.0207	3.8143 ± 0.1260	**4.76E+03**	**3.40E+03**	**5.36E+03**

techniques. For this purpose, we have generated two new VANET instances from the initial Urban scenario (of Malaga) by enlarging the metropolitan area considered. Therefore, as Figure 9.6 shows, the initial urban area (A1) has been expanded to A2 and A3 VANET areas. We have set the traffic flow as described in Section 9.5.1, also increasing the number of vehicles as follows:

- $Urban_{A1}$ with 30 vehicles in 120,000 m^2,
- $Urban_{A2}$ with 40 vehicles in 240,000 m^2,
- $Urban_{A3}$ with 50 vehicles in 360,000 m^2.

From the point of view of the mean fitness obtained by each algorithm (out of 30 independent runs), we can observe in Table 9.6 that PSO keeps the best performance for $Urban_{A2}$ and $Urban_{A3}$. Although, one of the most interesting results can be observed in GA, which arises as the second best algorithm in improving its behavior with the VANET size. ES obtains moderate mean fitness values for all network instances, keeping a low standard deviation. The worst results are registered by SA in $Urban_{A2}$, and DE in $Urban_{A3}$. Concerning DE, the initial choice of its parameters (Cr=0.9 and $\mu = 0.1$) could lead the algorithm to perform an exploitative search, hence obtaining good results in small instances (the second best for $Urban_{A1}$) but damaging its behavior in larger VANETs (the worst for $Urban_{A3}$). In summary, excepting for GA and DE, we can confirm that for the scaled VANET instances the performance of the algorithms are similar to their performances in $Urban_{A1}$ (the initial *Urban* VANET instance) being PSO always the best procedure.

Table 9.7: VDTP configurations and simulation output values for the optimal fitness achieved (in the median execution) by all studied algorithms. The last row contains the results obtained in the scope of the CARLINK project

Instance	Algorithm	VDTP Configuration			Simulation Results		
		Chunk S. (Bytes)	Retrans. Time (seconds)	Att.s	Trans. Time (seconds)	Lost Packs.	Data Trans. (kBytes)
Urban	PSO	41,358	10.00	3	**3.41**	**0.27**	1,024
	DE	28,278	6.00	9	3.59	0.63	1,024
	GA	31,196	3.83	9	3.61	**0.27**	1,024
	ES	23,433	10.00	8	3.50	**0.27**	1,024
	SA	19,756	6.43	3	4.22	0.36	1,024
	Human Exp.	25,600	8.00	8	4.24	1.60	1,024
Highway	PSO	29,257	6.42	9	**24.67**	3.18	1,024
	DE	19,810	6.91	8	27.66	3.45	1,024
	GA	34,542	9.54	10	26.96	2.72	1,024
	ES	38,490	8.15	12	33.99	3.36	1,024
	SA	32,002	8.21	4	25.43	**2.54**	1,024
	Human Exp.	25,600	10.00	10	33.08	3.27	1,024

A secondary but also interesting observation lies in the mean fitness values, which are in $Urban_{A2}$ lower than in $Urban_{A1}$. We suspect that, in spite of the larger dimension of $Urban_{A2}$, the proportion of communicating vehicles (per m^2) in this VANET helps the protocol operation specially for intermediate nodes, hence improving the effective ratio of delivery packets and the overall retransmission time. This proportion could not be enough for $Urban_{A3}$ where the cost of transmissions is the larger one.

Concerning the execution time, Table 9.6 shows in the three last columns the time required to find the best solution (T_{best}) for each VANET instance. Surprisingly, for PSO, ES, and SA the time required to converge in $Urban_{A2}$ is lower than in $Urban_{A1}$. This behavior can be explained by the fact of obtaining good solutions faster in $Urban_{A2}$ than in $Urban_{A1}$, where the lower number of vehicles could harm the communications conditions. On the contrary, the global run time (T_{run}) always increases with the network size. This is of course an expected result.

9.5.6 QoS Analysis

Finally, from the point of view of the worked solutions as VDTP configurations, we analyze the results in terms of the QoS indicators considered here: the transmission time, the number of lost packets, and the amount of data transferred induced in the designed VANET. In this sense, Table 9.7 shows the results after simulating the best solutions found by the studied algorithms. In addition, the last row of this table contains the results of simulating the configuration of VDTP that has been used in the scope of the CARLINK project (real word results with actual cars).

For the Urban VANET, the VDTP configuration obtained by PSO (Chunk_Size=41,358 Bytes, Retransmission_Time=10 s, and number of Attempts=3) achieves the best performance in terms of transmission time and mean number of lost packets. Specifically, in comparison with the human experts configuration of CARLINK, PSO obtains a reduction in the transmission time of 0.83 seconds (19.5%) registering also a lower number of lost packets. Nevertheless, it is in the Highway scenario were PSO obtains the higher time reduction of 8.41 seconds (25%) regarding the human experts configuration (from 33.08 s to 24.67 s). We must notice that, in spite of achieving the PSO a higher reduction in the transmission time than SA and GA, the fact of losing more packets (3.18 in PSO, 2.71 in GA and 2.54 in SA) in the global transference leads SA and GA to calculate a better fitness value (as shown in Table 9.3).

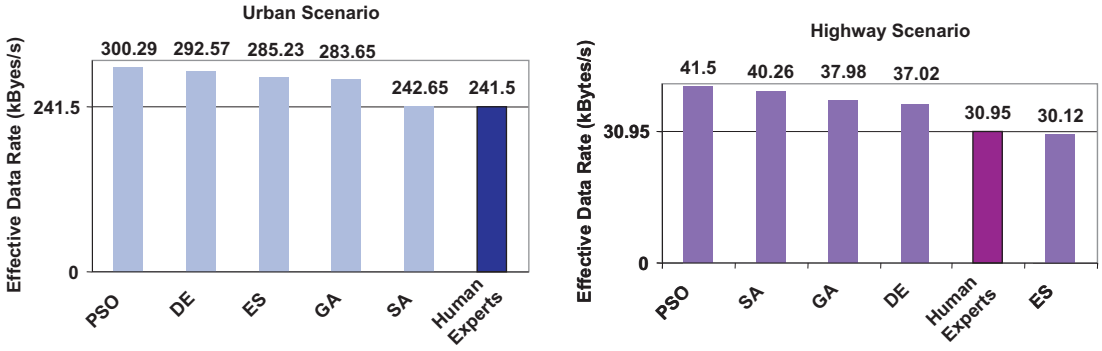


Figure 9.7: Effective transmission data rates (Throughput in kBytes/s) achieved during the simulations of final VDTP configurations in comparison with real values given by human experts' ones (CARLINK)

A final analysis can be done concerning one main QoS indicator: the *effective transmission data rate (throughput)*[6] achieved. As we can observe in Figure 9.7, the VDTP configuration obtained by practically all algorithms in the two VANET scenarios obtained higher effective data rates than the human configured VDTP. Specifically, PSO achieves the highest effective data rate (300.29 kBytes/s in Urban and 41.54 kBytes/s in Highway). This clearly claims for the utilization of these automatic algorithms to help human designers. We again remind that the actual correction of effective data rates between cars are in the order of tens of kBytes/s, so our savings (58.79 kBytes/s in Urban and 10.5 kBytes/s in Highway) are truly meaningful in current real applications such e.g. safety, traffic control, and weather predictions.

9.6 Conclusions

In this chapter, we have tackle the optimal File Transfer protocol Configuration (FTC) in VANETs by means of PSO, as well as other four popular metaheuristic algorithms. For this, we have designed a complex system accounting for a flexible simulation structure targeted for optimizing the transmission time, the number of lost packets, and the amount of data transferred in simulated and also realistic VANET scenarios.

The experiments, using *ns-2* (well-known VANET simulator), reveal that all algorithms are capable of efficiently solve the optimum FTC problem. Nevertheless, we have to highlight that PSO performs statistically better than all algorithms in Urban and statistically better than DE and ES in Highway. In addition, GA and SA show a competitive performance in Highway. The scalability analysis shows that GA improves with the network size, whereas DE decreases its performance with large VANET instances. PSO keeps the best result even for larger instances.

From the point of view of its real world utilization, PSO can reduce 19% of the transmission time in Urban and 25.43% in Highway with regards to human experts configuration of CARLINK, while transmitting the same amount of data (1,024 kBytes). The highest effective data rates

[6]In our fitness function, instead of using the *throughput* as extra control parameter, we have broken down it into the transmission time and data transferred directly in order to count them separately and enhance the search process of the algorithms.

obtained by PSO (of 300.39 kBytes/s in comparison with 241.5 kBytes/s of human experts) and DE (292.57 kBytes/s) in Urban lead us to advise the final use of our automatic design algorithms.

As a matter of further work we are presently extending our benchmark with new VANET realistic instances (e.g., complete cities and highway knots). In addition, we are planning to define new optimized configuration schemes for other communication protocols such as: UDP, DSR, etc. which should efficiently support actual VANET design.

Chapter 10

Optimal Signal Light Timing

10.1 Introduction

We now here go for our third real world problem addressed with techniques based in PSO and its swarm intelligence kind of search.

Pollution, congestion, security, parking, and many other problems derived from vehicular traffic are present every day in most cities around the world. Since changes in urban area infrastructure are usually not possible, researchers often agree in that a correct staging of signal (traffic) lights can help to reduce these problems by improving the flow of vehicles through the cities [MM10], [SGR08], [SC97]. Nevertheless, as signal lights are installed in cities and their number grows, their joint programming becomes more complex due to the huge number of combinations that appear, and hence, the necessity of automatic systems to optimally program the cycles of signal lights is beyond doubt.

In this sense, current research efforts in the field of automatic traffic control signals are directed to two main initiatives: on the one hand, automatic models of adaptation of signal control are designed [BHR96, HB01, ZDZ12] to change cycle program durations in every moment of the day as vehicles in queues demand those changes. The operation of these kind of tools is directly related to the sensor system and real time computation of traffic flow in real time. Although these tools demonstrate a successful performance in several cities around the world [BHR96, Bin01], the real management of the traffic network has a high cost of operation, although "real world" generally tends to repeat traffic flow patterns (rush hour, holidays, etc.).

On the other hand, modern simulators [HR04, KD10, LKH01] are very useful in helping the traffic management, since they provide researchers with an immediate and continuous source of information about the traffic flow. In addition, economical issues are also taken into account in these kinds of researches, since the use of real traffic tests implies the necessity of additional staff and sensoring platforms. Many studies in traffic flow simulation have been performed representing both macroscopic [MM10] and microscopic [SGR08], [TLTM05] traffic views. In the last few years, efforts have been concentrated on combining an accurate microscopic modeling [KD10], [SGR08] and the programming of convenient cycles [Nag10]. In this sense, the use of intelligent methods have demonstrated their usefulness to the optimization of programming cycles [ARG$^+$08], [SGR08]. However, authors in general have addressed specific urban areas with few intersections and a small number of signal lights (from 1 to 4 intersections with around 2 signal lights controlling each intersection) [BBSS01], and most of them consist on ad-hoc approaches only for one specific

instance [ARG+08], [SGR08]. The use of intelligent techniques for large and heterogeneous cases of study is still an open issue [Kut04]. It is a complex problem, since the greater the number of adjacent intersections, the higher the interaction between the signal lights, which increases the complexity of the problem by introducing a high epistasis between variables.

All this motivated us to propose a technique based on a Particle Swarm Optimizer [PCG11, KE01, GNAO12] that will be shown to find successful cycle programs of signal lights coupled with SUMO (Simulator of Urban Mobility) [KBW06], a well-known microscopic traffic simulator. Several features led us to use PSO instead of other optimization techniques: first, the PSO is a well-known algorithm shown to perform a fast converge to suitable solutions [Cle99]. This is a highly desirable property for the optimal cycle programming, where new immediate signal light schedules could be required to address updating events in traffic scenarios. Second, PSO is easy to implement, and requires few tuning parameters [Cle99]. Third, PSO is a kind of Swarm Intelligence algorithm that can inform us of future issues when dealing with this problem using independent agents in the system for online adaptation (a future line for us).

The task of SUMO is the evaluation of cycle programs (codified as vectors provided by our PSO) of the signal lights that control the scenario instance. In the present study, we have tested our proposal with two large and heterogeneous metropolitan areas with hundreds of signal lights located in the cities of Bahía Blanca in Argentina, and Malaga in Spain. In concrete, the main contributions that we can highlight in this chapter are the following ones:

- We propose a new PSO approach capable of obtaining efficient cycle programs for realistic urban scenarios. In this new approach, the initialization method, the solution encoding, the fitness function, and the velocity calculation have been adapted to deal with optimal cycle programs of signal lights.
- The behavior of our proposal is analyzed under different conditions of road network dimension and traffic density. An analysis of the computational effort is also made.
- In comparison with predefined cycle programs close to real ones, our PSO obtains quantitative improvements in terms of the two main objectives: the number of vehicles that reach their destinations and the global trip time.
- Further comparisons against other optimization methods (Random Search, Differential Evolution, and Standard PSO 2011) will justify the use of our PSO for the problem tackled.

The remainder of this chapter is organized as follows. In Section 10.2, a review of related works in the literature is presented. In Section 10.3, basic concepts of SUMO are given. In Section 10.4, our optimization technique proposal is described. Sections 10.5 and 10.6 present the experimental methodology used and the results obtained, respectively. Conclusions and future work are given in Section 10.7.

10.2 Literature Overview

There are different approaches in the state of the art that deal with signal light staging problems.Adaptive signal lights consider the "real" time impact of the traffic cycle duration over the traffic network. Several efforts have been done in this sense, mainly focused on use of detectors to sense the traffic and to change the duration of cycle programs, taking into account the actual flow of vehicles [BHR96, HB01, ZDZ12].

In this sense, several research studies employ a fuzzy part inside the intersection system control generally combined with other computational intelligence technique or heuristic [SCC06]. In [GS10], the authors adopted type-2 fuzzy set and designed a distributed multi-agent traffic-responsive signal control system. This system was tested on virtual road networks with several scenarios. Results showed superior performance of the approach in handling unplanned and planned incidents and obstructions. An adaptive traffic control model of signal lights is introduced by Bretherton et. al [BHR96] consisting on the SCOOT (Split Cycle Offset Optimization Technique) platform. SCOOT is an adaptive system for managing and controlling traffic signals in urban area, that responds automatically to fluctuations in traffic flow through the use of detectors on-street. This tool is especially useful for areas where traffic patterns are unpredictable.

Another adaptive method is UTOPIA (Urban Traffic Optimization by Integrated Automation) / SPOT (System for Priority and Optimization of Traffic) designed and developed by the FIAT Research Center [Woo93]. This system is aimed at improving both, private and public transport vehicles flow. UTOPIA/SPOT is a distributed real-time traffic-control system, especially suited for countries with advanced public transport services (tested in Italy, Norway, Netherlands, Sweden, Finland and Denmark). This system uses a hierarchical-decentralized control strategy, involving intelligent controllers to communicate with other signal controllers and with a central computer.

Different authors analyzed the use of fuzzy logic controllers at intersections of streets for adaptive tools. In an early study, Lim et al. [LKH01] proposed a fuzzy logic controller for real-time local optimization of only one intersection. Later, in Karakuzu et al. [KD10] a traffic simulator using fuzzy logic agents was developed for signal lights at isolated junctions. The results showed a minimization of the queue of vehicles on the roads, however their implementation is very compromised from an economic point of view, and the system's deployment required a great inversion. Other authors applying fuzzy logic were Rahman and Ratrout [RR09], with satisfactory results in a segment of the King Abdullah road in Saudi Arabia. The scenario shown in that paper was composed of four intersections and two signal lights for each one. An exhaustive review about automatic adaptive systems can be found in [HB01] and [ZDZ12].

Concerning the optimization strategy, we can find publications in which different resolution techniques have been applied: mathematic models, fuzzy logic approaches, and biologically inspired optimizers. Several authors employed mathematic techniques for tackling this kind of problem. For example, McCrea et al. [MM10] combined continuous calculus based models and knowledge-based models in order to describe the traffic flow in road networks. Tolba et al. [TLTM05] introduced a Petri Net based model to represent the traffic flow, from a macroscopic point of view (where only global variables are observed) and from a microscopic one (where the individual trajectories of vehicles are considered). More recently, Lammer and Helbing [LH08] designed a multi-agent traffic model inspired by the self-organizing fluctuations of vehicles in traffic jams. They used a simplistic simulation model considering only one direction of movement at a time. In Hawage et al. [HR04], the authors proposed a special-purpose simulation tool for optimizing traffic signal light timing. This tool provided complete traffic information, although it was limited to work only with four intersections.

Recently, biologically inspired techniques like Cellular Automata (CA) and Neural Networks (NN) have been used for tackling the underlying combinatorial optimization problems, and in particular for solving signal light staging problems. Brockfeld et al. [BBSS01] applied a CA model in which the city network was implemented as a simple square with a few normal streets and four intersections. Spall and Chin [SC97] presented a Neural Network for the configuration of control parameters in signal lights. In this approach, the vehicles needed an additional module for the data managing.

Related to the previous ones, metaheuristic algorithms [BR03] have become very popular for solving signal light staging problems. A first attempt was made by Rouphail et al. [RPS00], where a Genetic Algorithm (GA) was coupled with the CORSIM [HTSL07] microsimulator for the timing optimization of nine intersections in the city of Chicago (USA). The results, in terms of total queue size, were limited due to the delayed convergence behavior of the GA.

In Teklu et al. [TSW07], the impact of signal time changes with respect to the drivers was analyzed. More precisely, the authors considered the problem of determining optimum signal timings while anticipating the responses of drivers as an instance of the network design problem (NDP). An NDP aims to improve an existing network so that some total network performance measure is optimized with respect to some discrete or continuous design variables, while considering the user's reaction to the improvement. In order to solve the traffic equilibrium problem they used the SATURN (Simulation-Assignment Modeling package, [VV82]). The authors applied a macroscopic point of view of the traffic flow and employed a GA to compute the signal setting NDP (cycle time, offset, and green light times for stages). It is important to note that, the chromosome (grey-code) encoding was done differently for each particular instance under study.

In Sánchez et al. [SGR08], following the model proposed in Brockfeld et al. [BBSS01], the authors designed a GA with the objective of optimizing the cycle programming of signal lights. This GA was tested in a commercial area in the city of Santa Cruz de Tenerife (Spain). In this work, they considered that every intersection had independent cycles. As individual encoding, they used a similar binary (grey-code) representation to the one used in Teklu et al. [TSW07]. The computation of valid states was done before the algorithm began, and it strongly depended on the scenario instance tackled.

Turky et al. [TAYH09] used a GA to improve the performance of signal lights and pedestrians crossing control in a unique intersection with four-way two-lane. The algorithm solved the limitations of traditional fixed-time control for passing vehicles and pedestrians, and it employed a dynamic control system to monitor two sets of parameters.

A few publications related to the application of PSO for the schedule of signal lights also exist. In [CX06], authors applied a PSO for training a fuzzy logic controller located in each intersection by determining the effective time of green for each phase of the signal lights. A very simple network with two basic junctions was used for testing this PSO. Peng et al. [PWDL09] presented a PSO with isolation niches to the schedule of signal lights. In this approach, a custom microscopic view of the traffic flow was proposed for the evaluation of the solutions. A purely academic instance with a restrictive one-way road with two intersections was used to test the PSO. Nevertheless, this last study focused on the capacity of isolation niches to maintain the diversity of the swarm, and was not very concerned with the problem itself.

Finally, in Kachroudi and Bhouri [KB09] a multi-objective version of PSO is applied for optimizing cycle programs using a predictive model control based on a public transport progression model. In this work, private and public vehicles' models are used performing simulations on a virtual urban road network made up of 16 intersections intersections and 51 links. Each intersection is then controlled by a signal light with the same cycle time of 80 seconds.

All these approaches focused on different aspects of the signal light programming. However, three common features (limitations) can be found in all of them:

- They tackled limited vehicular networks with a few signal lights and a small number of other traffic elements (roads, intersections, directions, etc.). In contrast, our PSO can find optimized cycle programs for large scenarios with hundreds of signal lights, vehicles, and other elements.

- Almost all of them were designed for only one specific scenario. Some of them studied the influence of the traffic density. Our approach can be easily adapted to represent different scenario topologies. In this present study, we tackle two real scenarios with different combinations of signal lights and vehicles, fixing a number of 18 instances.

- They were not compared against other techniques. Our PSO is compared here against four different approaches: a Random Search algorithm, a Differential Evolution, the Standard PSO 2011, and the cycle program generator provided by SUMO.

10.3 SUMO: Simulator of Urban MObility

SUMO (Simulator of Urban Mobility) [KBW06], is a well-known traffic simulator that provides an open source, highly portable, and microscopic road traffic simulation tool designed to handle large road scenarios. SUMO requires several input files that contain information about the traffic and the streets to simulate. A network (*.net.xml* file) holds the information about the structure of the map: nodes, edges, and connections between them. The network can be imported from popular digital maps such as OpenStreetMap [HW08] and converted to a valid SUMO network by means of a series of scripts provided in the SUMO package. We have chosen OpenStreetMap (OSM) because it provides both, geographic data and signal light information.

A journey is a vehicle movement from one location to another defined by: the starting edge (street), the destination edge, and the departure time. A route is an extended journey, meaning that, a route definition contains not only the first and the last edges, but also all the edges the vehicle will pass through. These routes are stored in a demand file (*.rou.xml* file) either through a route generator given by SUMO, existing routes imported from other software, or by hand. Additional files (*.add.xml*) can add to SUMO information about the map or about the signal lights. SUMO allows to replace and edit information on the cycles of signal lights by manipulating a file with *.add.xml* extension. It is important to note that SUMO provides by default the valid combination of states that the signal lights controller can go through inside the map specification file (*.net.xml* file) [KBW06], and an approximation of interval times for these states [LKA04]. This means that SUMO already incorporates a solver algorithm for the cycle program of signal lights based on greedy and human knowledge. That solver will be called here SCPG (SUMO Cycle Program Generator) and it will be used in a comparison against our PSO.

The output of a SUMO simulation is registered in a trip information file (*.tripinfo.xml*) that contains information about each vehicle's departure time, the time the vehicle waited to start at (offset), the time the vehicle has arrived, the duration of its journey, and the number of steps in which the vehicle speed was below 0.1m/s (temporal stops in driving). This information is used for the evaluation of cycle programs of signal lights.

10.3.1 SUMO Data Structure

As previously mentioned, the main objective of our approach is to find optimized cycle programs (duration of color states of signal lights) for all the signal lights located in a given urban area. At the same time, these programs have to coordinate signal lights in adjacent intersections with the aim of improving the global flow of vehicles circulating within the established routes. For this reason, we have focused on a microscopic view of the management of traffic agents but, at the same time, we want to evaluate the behavior of all the vehicles in the complete urban scenario during a given time span.

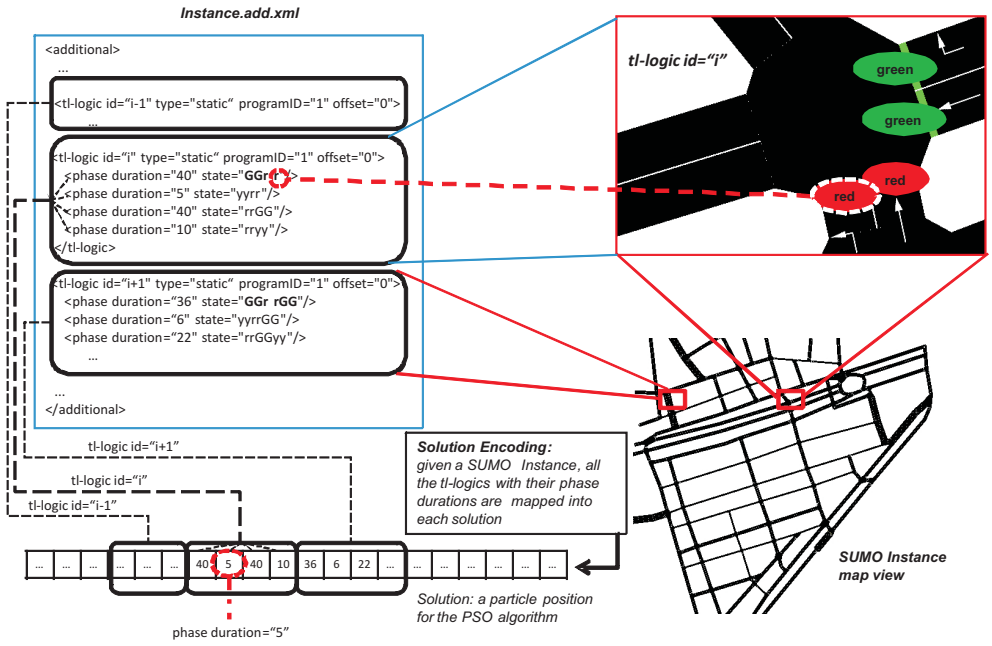


Figure 10.1: Intersection with four signal lights selected from the SUMO instance map. Phase durations (cycles) are specified in *Instance.add.xml* file and encoded inside a PSO tentative solution

The evaluation of the resulting signal light timing programs is carried out by means of automatic simulations. For this task we use SUMO. The simulation structure of SUMO is comprised of a series of elements that we have taken into account for developing our traffic scenarios. A SUMO instance for a urban traffic scenario is basically composed of: intersections, signal lights, roads, and vehicles moving through their previously specified routes. The signal lights are located in intersections (junctions in SUMO), and control the flow of vehicles by following their programs of color states and cycle durations. In this context, all signal lights located in the same intersection are governed by a common program, since they have to be necessarily synchronized for traffic security. In addition, for all the signal lights in an intersection, the combination of color states during a cycle period is always kept valid [LKA04] and must follow the specific traffic rules of intersections, in order to avoid vehicle collisions and accidents. For example, two signal lights located in the same intersection but controlling conflicting movements must not be in green during the same time instance. In this sense, as illustrated in Figure 10.1, SUMO provides a complete set of valid combinations of color states for each intersection, which can not be modified during the optimization process. This avoids invalid combinations of color states and restricts the optimization approach to work only with feasible states.

Figure 10.1 shows an illustration of the main elements constituting the cycle program of signal lights in SUMO. This program staging is implemented in an XML file (`instance.add.xml`) that SUMO uses for loading cycles and states, previous to the simulation process. In this file, each `tl-logic` element corresponds to an intersection. Following the model designed by Krajzewicz et al. [KBW06], a `tl-logic` comprises a sequence of `phases` in a cyclic way during the simulation time. Each phase indicates the corresponding colors states (attribute `state`) of all the signal lights in the intersection, and the duration of this state (attribute `duration`).

An example of this mechanism can be observed in Figure 10.1 where the `tl-logic` with `id="i"`, that corresponds to an intersection of the SUMO instance, contains four phases with durations of 40, 5, 40, and 10 seconds (simulation steps). In these phases, the states have four colors, corresponding each one of them to one of the four signal lights located in the studied intersection. These states are the valid ones generated by SUMO adhering to real traffic rules. In this instance, the first phase contains the state *GGrr* meaning that two signal lights are in green (G), and the other two are in red (r) during 40 seconds. The following phase changes the state of the four traffic lights to *yyrr* (y is amber) during 5 seconds, and so on. The last phase is followed by the first one, and this cycle is repeated throughout the simulation time. All the `tl-logics` in the complete SUMO instance perform their own programming cycles of phases at the same time, hence constituting the global staging of signal lights. Therefore, programming cycles are the main focus of this work, since we are interested in optimizing the combination of phase durations of all signal lights (in all intersections) with the aim of improving the global flow of vehicles circulating in a urban scenario instance.

A final indication in this sense concerns the behavior of the vehicles involved in the SUMO instance scenario, that depends on both road directions and speed. SUMO employs a space-discrete extended model as introduced by Krauß et al. [Kra98]. In this model, the streets are divided into cells and the vehicles circulating through the streets go from one cell to another if allowed. The speed of each vehicle depends on its distance from the vehicle in front of it, with a preestablished maximum speed typical of urban areas (50 km/h in our study).

10.4 PSO for Traffic Light Scheduling

This section describes our optimization solver proposed for the optimal cycle programs of signal lights. It describes the solution encoding, the fitness function, and finally the global optimization procedure.

10.4.1 Solution Encoding

Following the structure of programming cycles adopted by SUMO, the global staging of signal lights has been easily encoded by means of a vector of integers, where each element represents a phase duration of one state of the traffic lights involved in a given intersection. This way, as shown in Figure 10.1, all the phase durations in the `tl-logic` elements are successively placed in the solution vector, hence mapping the complete staging of signal lights in a simple array of integers. The reason of working with this representation is twofold: first, the SUMO simulator works itself with integer values for representing the discrete sequence of time steps (seconds) that make up the complete simulation procedure. Second, real signal lights also employ integer values for specifying the duration of phases in their internal programs.

In spite of its simplicity, this solution representation allows our PSO to take into account the interdependency of variables, not only between phase durations in a common `tl-logic` element, but also between signal lights at adjacent intersections. In this sense, PSO is known to have a successful performance with non-separable problems [Cle99, HRM+08], which is the case in this approach. This last is an interesting feature since solutions with coordinated signal lights (located in different but close intersections) could be then promoted by our optimization algorithm.

10.4.2 Fitness Function

Each solution vector (s), codifying the cycle program of the signal light programs, is evaluated considering the information obtained from the events happening during the simulation by means of the following equation:

$$fitness(s) = \frac{\left(\sum_{v=0}^{V} j_v(s)\right) + \left(\sum_{v=0}^{V+C} w_v(s)\right) + (C(s) \cdot St)}{V^2(s) + Cr} \tag{10.1}$$

The main objective consists of maximizing the number of vehicles that reach their destination (V) during the simulation time (St). Namely, minimizing the number of vehicles that do not reach their destination and remain circulating (C) after the simulation time is reached. A secondary but important objective is to minimize the global duration of the vehicle's journeys (j_v). It is clear that the global duration concerns the journey time of the vehicles that reach their destination during the simulation process. On the contrary, vehicles with incomplete journeys (C) consume all the simulation time St and then, an additional penalization is induced by multiplying these two factors. It is worth mentioning that terms in Equation 10.1 are in the range of values $[1e+0 \cdots 5e+2]$ and therefore, additional weighting values were not considered in this formulation. Only, the number of vehicles that arrive at their destinations is squared (V^2) in order to prioritize it over the other terms and factors.

An important factor concerns the state of the signal lights in each precise moment, since it influences the time that each vehicle must stop and wait (w_v), with the consequent delay on its own journey time, e.g., a prolonged state of signal lights in red could collapse the intersection where it is, and even close other intersections. However, a prolonged state in green could improve the traffic flow in a given area or direction, but also makes the traffic flow of other areas and directions worse. In this sense, a balanced number of color lights in the phase duration of the states should promote those states with more signal lights in green located in streets with a high number of vehicles circulating, and signal lights in red located in streets with a low number of vehicles moving. The ratio of colors in each phase state of all the tl-logic tl (intersections) can be formulated as follows:

$$Cr = \sum_{k=0}^{tl} \sum_{h=0}^{ph} s_{k,h} \cdot \left(\frac{G_{k,h}}{R_{k,h}}\right), \tag{10.2}$$

where $G_{k,h}$ is the number of traffic lights in green (G), and $R_{k,h}$ is number of signal lights in red in the phase state h (with duration $s_{k,h}$) and in the tl-logic k. The minimum value of $r_{k,h}$ is 1 in order to avoid division by 0.

10.4.3 Optimization Strategy

Our optimization strategy is composed of basically two main parts: an optimization algorithm and a simulation procedure. The optimization part is carried out by means of the Particle Swarm Optimization algorithm which has been specially adapted to find optimal (or quasi-optimal) cycle programs for signal lights. It works as follows:

1. The *initial swarm* is composed of a number of particles (solutions) initialized with a set of random values representing the phase durations. These values are within the time interval $[5, 60] \in Z^+$, and constitute the range of possible time spans (in seconds) a signal light can

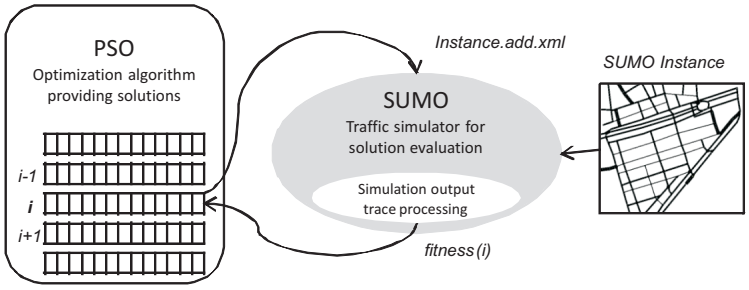


Figure 10.2: Optimization strategy for the cycle program configuration of signal lights. The algorithm invokes *SUMO* for each solution evaluation

be kept on a signal color (only green or red, the time for amber is a constant value). We have specified this interval by following several examples of real signal light programs provided by the City Council of Malaga (Spain).

2. The *velocity calculation* has been softly modified in order to deal with integer combinatorial values by truncating (with floor $\lfloor . \rfloor$ and ceiling $\lceil . \rceil$ functions) all elements (j) of the new velocity vector as Equation 10.3 shows:

$$v_i^{t+1}(j) = \begin{cases} \lfloor v_i^{t+\frac{1}{2}}(j) \rfloor & if \;\; U(0,1)_i^t(j) \leq \lambda \\ \lceil v_i^{t+\frac{1}{2}}(j) \rceil & otherwise \end{cases} \tag{10.3}$$

In this formula, $v_i^{t+\frac{1}{2}}$ is the intermediate velocity value obtained from Equation 3.2. The parameter λ determines the probability of performing ceil or floor functions in the velocity calculation ($\lambda = 0.5$ for this study).

3. The *inertia weight* changes linearly through the optimization process by using the following equation:

$$\omega = \omega_{max} - \frac{(\omega_{max} - \omega_{min}) \cdot g}{g_{total}} \tag{10.4}$$

This way, at the beginning of the process a high inertia (ω_{max}) value is introduced, which decreases until reaching its lowest value (ω_{min}). A high inertia value provides the algorithm with exploration capability and a low inertia promotes exploitation.

The simulation procedure is the way of assigning a quantitative quality value (fitness) to the solutions, thus leading to optimized cycle programs tailored to a given urban scenario instance. This procedure is carried out by means of the SUMO traffic simulator, which accepts new cycle programs of signal lights and computes the required values in Equation 10.1.

As Figure 10.2 illustrates, when PSO generates a new solution it is immediately used for updating the cycle program. Then, SUMO is started to simulate the scenario instance with streets, directions, obstacles, signal lights, vehicles, speed, routes, etc., under the new defined staging of cycle programs. After the simulation, SUMO returns the global information necessary to compute the *fitness* function. Each solution evaluation requires only one simulation procedure since vehicle routes in SUMO were generated deterministically. In fact, as suggested in [SGR05], stochastic traffic simulators obtain similar results to deterministic ones, the latter allowing huge computing savings.

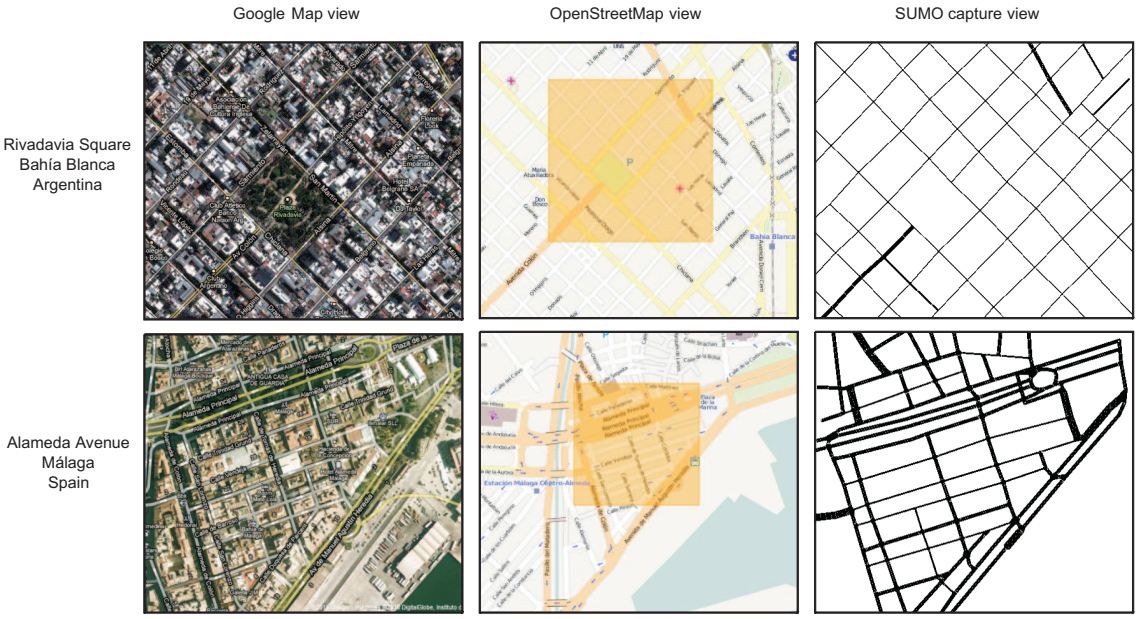


Figure 10.3: Process of creation of real-world instances for study. Rivadavia Square (38°43'03"S 62°15'56"O) and Alameda Avenue (36°43'60"N 4°25'87"O) instance views. After selecting our area of interest (Google map view), it is interpreted by means of the OpenStreetMap tool, and then exported to SUMO in XML format

In addition, we have to notice that each new cycle program is statically loaded for each simulation procedure. Our aim here is not to generate cycle programs dynamically during an isolated simulation as done in agent-based algorithms [KBM+05], but obtaining optimized cycle programs for a given scenario and timetable. In fact, what real signal light schedulers actually demand are constant cycle programs for specific areas and for preestablished time periods (rush hours, nocturne periods, etc.), which led us to take this focus.

10.5 Methodology of Our Study

This section presents the experimental framework followed to assess the performance of our optimization solver. First, we describe the signal light scenario instances generated specifically for this work. Later, the implementation details and parameter settings are presented.

10.5.1 Instances

As we are interested in developing an optimization solver capable of dealing with close-to-reality and generic urban areas, we have generated two scenarios by extracting actual information from real digital maps. These two scenarios cover similar areas of approximately $0.42km^2$, and are physically located in the cities of Bahía Blanca in Argentina, and Malaga in Spain. The information used concerns: traffic rules, traffic element locations, buildings, road directions, streets, intersections, etc. Moreover, we have set the number of vehicles circulating, as well as their speeds

Table 10.1: Rivadavia Square and Alameda Avenue instances' specifications

City Instance	Number of Traffic Logics	Number of Traffic Lights	Number of vehicles
Rivadavia Square (Bahía Blanca)	20	88	100 300 500
	30	136	100 300 500
	40	176	100 300 500
Alameda Avenue (Malaga)	20	78	100 300 500
	30	130	100 300 500
	40	184	100 300 500

by following current specifications available in the Mobility Delegation of the City Hall of Malaga (http://movilidad.malaga.eu/). This information was collected from sensorized points in certain streets obtaining a measure of traffic density in several time intervals. In the case of Bahía Blanca we could not obtain such an information, and hence we considered the same number of vehicles as used for the Malaga scenario.

In Figure 10.3, the selected areas of the two cities are shown with their corresponding capture views of OpenStreetMap and SUMO (as explained in Section 10.3.1). Other driving styles such as the Commonwealth/British one could be also tackled with our approach, since we can easily capture with OpenStreetMap and export to SUMO areas of UK cities, and work with them by following their directions and traffic rules. The specific features of the selected areas are as follows:

1. *Rivadavia Square*. Located in the city center of Bahía Blanca (Figure 10.3, top), it is made up of 53 intersections between streets that form a practically regular grid of blocks, as is usual in American cities. Except for the main avenue, almost all streets are one way with opposite directions to each others. Therefore, the great majority of traffic logics (junctions) in this scenario have four signal lights: straight on, left, and the two ones on the perpendicular street.

2. *Alameda Avenue*. The city center of Malaga (Figure 10.3, bottom draw), represents the common irregular structure of European cities, having different street widths and lengths. It is composed of 73 junctions between streets and roundabouts. Each traffic logic in this scenario includes from 4 to 16 signal lights.

We have considered these two scenarios since they constitute quite different urban areas with heterogeneous structures and traffic organizations. Moreover, with the aim of obtaining generalized concluding results, the number of instances used in the experimentation has been increased by incorporating different numbers of vehicles moving through these streets, and different numbers of signal lights operating within the selected areas. Table 10.1 contains the combination of traffic logics and vehicles used in each instance for each scenario, constituting a total number of 18 instances: 9 for Rivadavia Square and 9 for Alameda Avenue. We have to notice that in spite of both scenarios having similar scales of traffic logics (20, 30, and 40), the number of signal lights is not the same, since they contain different intersection shapes.

Table 10.2: Simulation and PSO parameters

Solver Phase	Parameter	Value
Simulation Details	Simulation Time (steps)	500 sec.
	Simulation Area	0.45 km^2
	Number of Vehicles	100/300/500
	Vehicle Speed	0-50 km/h
	N. of Traffic Logics	20/30/40
PSO Parameters	Max. N. of Evaluations	30,000
	Swarm Size	100
	Particle Size (N. Traffic Lights)	88/136/176 78/130/184
	Local Coefficient (φ_1)	2.05
	Social Coefficient (φ_2)	2.05
	Maximum Inertia (w_{max})	0.5
	Minimum Inertia (w_{min})	0.1
	Velocity Truncation Factor (λ)	0.5

Concerning the number of vehicles, we have considered three different scales of 100, 300, and 500 cars for each instance (as shown in Table 10.1) circulating throughout the simulation time. Each one of the vehicles performs its own route from origin to destination circulating with a maximum speed of 50 km/h (typical in urban areas). The routes were previously generated by following random paths and covering as much as possible all network entries. Starting times of vehicles were also uniform randomly specified throughout the whole simulation. This means that, at the same time, only a subset of the whole set of vehicles is circulating through the network. The simulation time was fixed to 500 seconds (iterations of microsimulation) for each instance. This time was determined as a maximum time for a car to complete its route, even if it must stop at all the signal lights it comes across. When a vehicle leaves the scenario network, it will not appear again.

10.5.2 Experimental Setup

We have used the implementation of the PSO algorithm provided by the MALLBA library[ALGN+07]. The simulation phase is carried out by executing (in the evaluation of particles) the traffic simulator SUMO release 0.12.0 for Linux. The experiments were performed in computers at the laboratories of the Department of Computer Science of the University of Malaga (Spain). Most of them are equipped with modern dual core processors, 1GB RAM, and Linux Debian O.S. They operate under a Condor [TTL05] middleware platform that acts as a distributed task scheduler (each task dealing with one independent run of PSO).

For each scenario instance we have carried out 30 independent runs of our PSO. The swarm (population) size was set to 100 particles performing 300 iteration steps, hence resulting in 30,000 solution evaluations (SUMO simulations) per run and instance. The choice of these two parameters (swarm size and maximum iteration steps) corresponds to previous tuning experiments as described in Section 10.6.1. The particle size directly depends on the number of signal lights of each instance (shown in Table 10.1). The remaining parameters are summarized in Table 10.2. These parameters were set after preliminary executions of PSO with the smallest instances of Rivadavia Square and Alameda Avenue (with 20 traffic logics and 100 vehicles). Specific parameters of PSO were selected as recommended in the studies about the convergence behavior of this algorithm in [Cle99] and [ES00]. According to these, acceleration coefficients φ_1 and φ_2 were set to 2.05 and inertia weight decreases linearly along with the increment of the iteration steps from 0.5 to 0.1.

Additionally, we have implemented three algorithms also in the scope of the MALLBA [ALGN+07] library, with the aim of establishing comparisons against our PSO. These three algorithms are a Random Search (RANDOM), a Differential Evolution (DE), and the Standard PSO 2011 (SPSO2011).

Algorithm 11 Pseudocode of RANDOM

```
1: x ← initializeSolution()
2: t ← 0
3: while t < MAXIMUM_t do
4:    generate(x^t)    //new solution
5:    if f(x) ≥ f(x^t) then
6:       x ← x^t
7:    end if
8:    t ← t + 1
9: end while
```

Thus, performing the same experimentation procedure, we expect to obtain some insights into the power of our proposal (how much intelligent it is) regarding a technique without any heuristic information in its operation (RANDOM), and with regards to two other difference-vector based metaheuristics: DE, and SPSO2011. In the case of SPSO2011, it is the last PSO proposal in [PCG11] and uses a different quantisation/discretization method to our PSO. The maximum number of evaluations was set to $30,000$, as for PSO.

The pseudocode of the Random Search algorithm is shown in Algorithm 11. It basically performs by keeping just the best solution found so far in the optimization procedure. For the sake of a fair comparison, we also adapted the DE for dealing with integer values in the solution codification, that is, using the same mechanism of ceiling/flooring ($\lceil . \rceil / \lfloor . \rfloor$) functions as done in the velocity vector calculation of PSO (Equation 10.3).

$$w_i^{t+1}(j) = \begin{cases} \lfloor w_i^{t+\frac{1}{2}}(j) \rfloor & if \;\; U(0,1)_i^t(j) \leq \lambda \\ \lceil w_i^{t+\frac{1}{2}}(j) \rceil & otherwise \end{cases} \tag{10.5}$$

In the case of DE, the truncation method is applied to the mutant vector w_i^{t+1}, as specified in Equation 10.5, also with $\lambda = 0.5$.Finally, as previously commented on in Section 10.3.1, SUMO provides a deterministic algorithm for generating cycle programs (SCPG, SUMO Cycle Programs Generator). Then we also compare the cycle programs obtained by our PSO against these ones obtained by SUMO. This last algorithm basically consists in assigning to the phase durations of the traffic logics fresh values in the range of [6,31], according to three factors:

1. the proportion of green states in the phases,
2. the number of incoming lanes entering the intersection, and
3. the braking time of the vehicles approaching the signal lights.

Further information on this algorithm can be found in [KBW06].

10.6 Analysis and Discussion of Results

Results and their analyses are presented in this section from several points of view. First, we study the performance of our optimization solver in comparison with other techniques, and its ability to report successful cycle programs for the different instances. After this, we present a brief report on the computational effort required for the experiments. Later, we turn the focus on the problem domain, and we examine representative reported solutions with the aim of justifying the use of our PSO with a potentially truly positive impact on traffic flow.

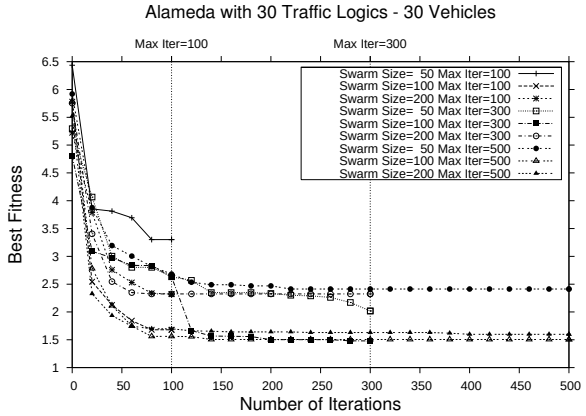


Figure 10.4: Traces of progress of the best fitness values (Median out of 30 independent runs) of PSO tackling with the Alameda Avenue instance with 30 traffic logics and 300 vehicles. The traces correspond to different configurations of swarm sizes (with 50, 100, and 200 particles) and maximum number of iterations (100, 300, and 500) as stop condition

10.6.1 Performance Analysis of Algorithms

Before any comparison takes place, we first wish to show a representative view of the internal behavior of our PSO under different conditions of swarm size and maximum number of iterations. We used this investigation as a basis for setting the most convenient values in the following experimentation. So, Figure 10.4 plots the traces of progress of the best fitness values (Median run out of 30 independent executions) of PSO tackling with the Alameda Avenue instance with 30 traffic logics and 300 vehicles. These traces correspond to different configurations of *swarm size* (Ss) with: 50, 100, and 200 particles, and *maximum number of iterations* ($MAXIMUM_t$) with: 100, 300, and 500 steps to the stop condition. It is worth saying that the number of iteration steps directly influences on the inertia weigh (in the velocity calculation of PSO), and hence, this parameter should be studied separately in combination with all different values of swarm size.

As shown in Figure 10.4, for almost all the combinations (of Ss and $MAXIMUM_t$) our PSO got to converge on the interval of 100 and 300 iterations, showing the combination of 100 particles in the swarm and 300 iteration steps the best performance results. In fact, for this configuration the fitness clearly improved after 100 iterations to finally converge just before 200 iteration steps (20,000 function evaluations). We have to mention that other configurations of PSO with $Ss = 200$ $MAXIMUM_t = 500$ also obtained such a successful results although requiring a higher computational cost with more than 50,000 function evaluations ($Ss = 100$ and $MAXIMUM_t = 500$), in contrast with 30,000 ones in the case of $Ss = 100$ and $MAXIMUM_t = 300$. Therefore, we opted to set 100 particles in the swarm and a maximum of 300 iteration steps in our experimentation.

From another point of view, Figure 10.5 plots the trace progress of the best fitness values obtained in 30 independent runs of PSO when solving the Rivadavia Square instance with 40 traffic logics and 500 vehicles. In this figure, we can observe that for all executions our algorithm practically converged after the first 150 iterations, using the remaining time to only slightly refine solutions. In addition, all the computed solutions are close each other in quality, but different among them. These are desirable features in terms of convergence and robustness, since we can offer to the expert a varied set of accurate cycle programs in a reduced time.

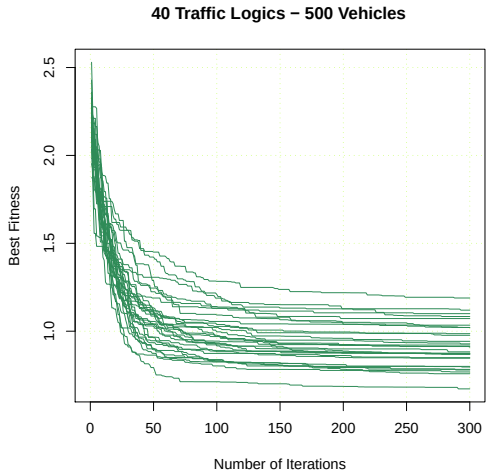


Figure 10.5: Trace progress of the best fitness values in 30 independent runs of PSO tackling with Rivadavia Square instance with 40 traffic logics and 500 vehicles

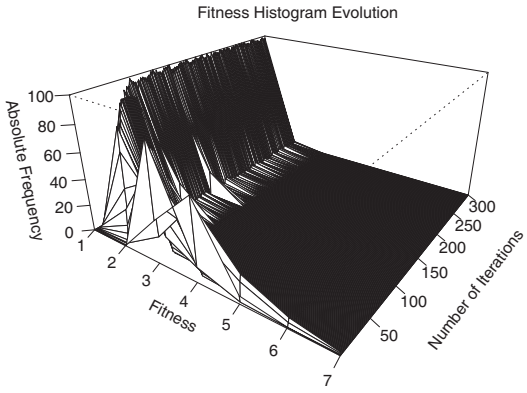


Figure 10.6: Swarm fitness histogram through 300 iterations in the optimization of the Rivadavia Square scenario with 40 traffic logics and 500 vehicles

To better explain this, Figure 10.6 plots the absolute frequency of the fitness distribution of the entire swarm through the optimization process of one typical execution. Specifically, it illustrates one of the thirty independent runs of our PSO tackling the Rivadavia Square scenario with 40 traffic logics and 500 vehicles. We can see that the initial particles are diverse and with high cost values ($\simeq 7$), although they were able to converge in a low fitness region (≤ 1) during the second half of the execution process. In this specific run, 475 vehicles out of 500 reached their particular destinations (95%) in a simulation time lower than 500 seconds (the complete number of microsimulation steps). This accurate behavior is also found in all executions and for all instances, and it represents another interesting feature of our approach.

Table 10.3: Median fitness values obtained by PSO for all the scenario instances. Median fitness obtained by RANDOM and by SCPG algorithms are also provided. NTL is the Number of Traffic Logics

Instance	NTL	Number of Vehicles								
		100			300			500		
		PSO	RANDOM	SCPG	PSO	RANDOM	SCPG	PSO	RANDOM	SCPG
Rivadavia Square	20	**1.64E+00**	2.91E+00	2.38E+00	**8.40E-01**	1.45E+00	9.24E-01	**7.93E-01**	1.51E+00	9.56E-01
	30	**1.80E+00**	3.11E+00	2.45E+00	**9.09E-01**	1.65E+00	9.57E-01	**8.79E-01**	1.72E+00	9.89E-01
	40	**1.79E+00**	3.08E+00	2.49E+00	**9.11E-01**	1.75E+00	9.76E-01	**8.96E-01**	1.74E+00	9.93E-01
Alameda Avenue	20	**9.47E-01**	1.68E+00	1.49E+00	**8.44E-01**	1.62E+00	1.29E+00	**4.10E+00**	7.87E+00	2.35E+01
	30	**1.56E+00**	3.55E+00	5.12E+00	**1.74E+00**	4.52E+00	6.00E+00	**7.67E+00**	1.33E+01	3.31E+01
	40	**1.88E+00**	3.98E+00	5.38E+00	**2.87E+00**	7.33E+00	1.83E+01	**9.39E+00**	1.64E+01	1.47E+01

Table 10.4: Median fitness values obtained by our PSO, DE, and Standard PSO 2011 for all the scenario instances. NTL is the Number of Traffic Logics

Instance	NTL	Number of Vehicles								
		100			300			500		
		PSO	DE	SPSO2011	PSO	DE	SPSO2011	PSO	DE	SPSO2011
Rivadavia Square	20	**1.64E+00**	2.18E+00	1.87E+00	**8.40E-01**	9.94E-01	9.82E-01	**7.93E-01**	9.80E-01	1.22E+00
	30	**1.80E+00**	2.25E+00	2.33E+00	**9.09E-01**	1.11E+00	1.28E+00	**8.79E-01**	1.02E+00	1.44E+00
	40	**1.79E+00**	2.23E+00	2.50E+00	**9.11E-01**	1.13E+00	1.25E+00	**8.96E-01**	1.10E+00	1.40E+00
Alameda Avenue	20	**9.47E-01**	1.22E+00	1.11E+00	**8.44E-01**	1.07E+00	9.12E-01	**4.10E+00**	4.98E+00	4.71E+00
	30	**1.56E+00**	2.19E+00	2.49E+00	**1.74E+00**	2.54E+00	3.47E+00	**7.67E+00**	8.57E+00	1.11E+01
	40	**1.88E+00**	2.54E+00	3.21E+00	**2.87E+00**	4.06E+00	5.32E+00	**9.39E+00**	1.17E+01	1.30E+01

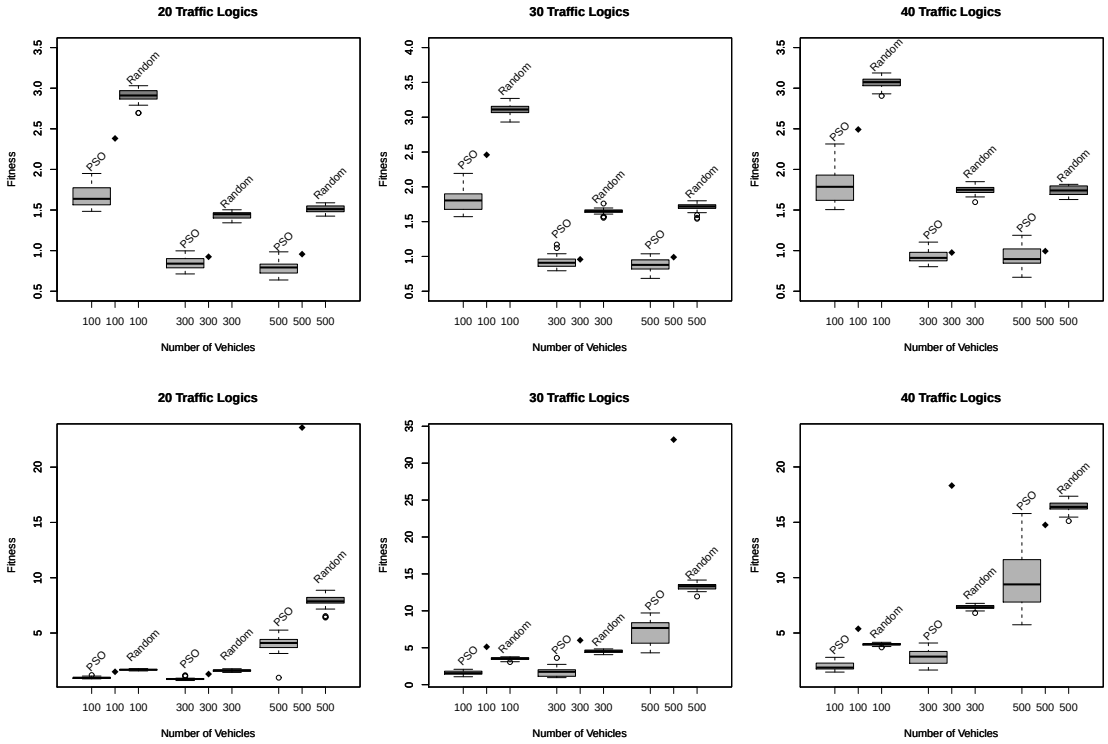


Figure 10.7: Boxplots representation of distributions results of Rivadavia Square (three at the top) and Alameda Avenue (three at the bottom) instances with 20, 30, and 40 traffic logics, and 100, 300, and 500 vehicles. The results of SCPG are represented with a ♦ point since this technique always returns the same deterministic result for a given instance

Table 10.3 contains the median fitness values obtained by the proposed PSO for all the scenario instances. Additionally, the median fitness values obtained by the RANDOM algorithm, and the results of the SCPG are also provided in order to allow comparisons. We can easily check in this table that PSO obtained the best median fitness (marked in bold) independently of both, the number of vehicles and the number of traffic logics in each scenario instance.

In order to provide statistically meaningful comparisons, we have applied a *Signed Ranking* (*Wilcoxon*) test [Wil87] to the numerical distributions of the results. We have used this non-parametric test since the resulting distributions usually violated the condition of normality required to parametric tests (Z *Kolmogorov-Smirnov* = 0.04). Another implication of the violation of the normality condition is the use of median values (as shown in Table 10.3) instead of other measures like the mean and the standard deviation [She07]. The confidence level was set to 95%, which allows us to ensure that all these results are statistically different if they result in *p-value*<0.05.

In effect, for all the instances, the differences between the distributions out of 30 independent runs resulted with *p-values* << 0.05. In general, the differences in the distributions of the medians (Table 10.3) resulted in a global *p-value* of 5.73E-7 when comparing PSO with RANDOM, and a global *p-value* of 6.33E-5 when comparing PSO with SUMO. Therefore, we can claim that our

PSO obtained statistically better results than the other two compared algorithms: RANDOM (stochastic search) and SCPG (deterministic). This also means that our algorithm is intelligent and competent when compared to greedy information and human knowledge.

A summary of these results can be seen in Figure 10.7, where the boxplots of the distribution fitness of PSO, and RANDOM are plotted. The results of SCPG are represented with a ♦ point since this technique always returns the same deterministic result. As expected, the distributions of PSO show better lower quartiles, medians, and upper quartiles than RANDOM for all the instances. Regarding SCPG, we can notice that the median values of PSO are in general better than the results of SCPG. Only in the case of Rivadavia Square with high densities of traffic (300 and 500 vehicles), the SUMO results get close to the upper quartiles of our PSO distributions.

Concerning the two scenario instances, the resulting fitness values in Rivadavia Square are in general better than the ones obtained in Alameda Avenue. This difference in the results is more noticeable when a large number of vehicles is circulating (500), where the median fitness differ in two orders of magnitude (from 7.93E-01 to 3.31E+01). We suspect that the regular structure of Rivadavia Square (see Figure 10.3) makes the traffic more fluid in this scenario than in Alameda Avenua (with irregular European design), which could lead the PSO to obtain different ranges of results in similar conditions.

10.6.2 Comparison with Other Metaheuristics

For a further comparison, we have studied the performance of two other metaheuristic algorithms for the same experimental procedure as with our proposal. A first comparison concerns a Differential Evolution algorithm (as described in Section 10.5.2), by means of which we expect to better justify the use of PSO on the cycle program of signal lights. Secondly, we compare our PSO against the Standard PSO 2011 which performs a different velocity calculation and discretization method.

The median fitness values (out of 30 independent runs) resulted in the experimentation of DE and SPSO2011 are included in Table 10.4 together with the ones of our PSO for the two scenario instances, Rivadavia Square and Alameda Avenue. Again, we confirm that the PSO obtained the best median fitness for all the combinations of number of vehicles and number of traffic logics in each scenario instance. In general, using a Wilcoxon Signed Rank test with α=0.05, the differences in the distributions of the medians (Table 10.4) resulted in a global *p-value* of 1.94E-4 when comparing PSO with DE, and a global *p-value* of 1.96E-4 when comparing PSO with SPSO2011. In the first case, the different learning procedures that our PSO and DE perform is the main factor that influences the statistical differences in results, since these two algorithms used the same discretization method. In the second case, the different velocity calculation methods influence the algorithms' performances of our PSO and SPSO 2011, indicating that our proposal is better than the last Standard PSO for the tackled problem.

As a further comparison, SPSO2011 showed better fitness values than DE, resulting a global *p-value* of 1.47E-2. If we take into account that DE uses a similar discretization method as to our PSO, the last results lead us to suspect that the different discretization of vectors marginally influences on the global algorithm's performance.

Therefore, in the scope of the experimental framework adopted in this approach, we can claim that our PSO also obtained statistically better results than the other metaheuristic approaches (DE and SPSO2011) used to solve the optimal cycle program of signal lights.

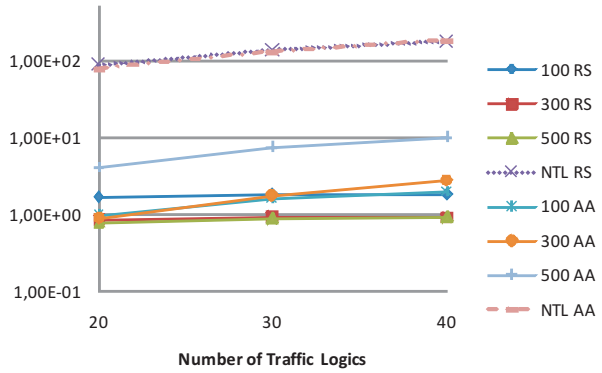


Figure 10.8: Increment of the median fitness with regards to the number of signal lights for the Alameda Avenue scenario. The values are in logarithmic scale

Table 10.5: Mean time and standard deviation in seconds of our PSO to compute all the experiments

Instance	Number of Traffic Logics	Number of Vehicles		
		100	300	500
Rivadavia Square	20	4.14E+02±6.74E+01	5.94E+02±6.83E+01	7.25E+02±6.53E+01
	30	4.09E+02±6.11E+01	5.04E+02±5.54E+01	7.44E+02±6.51E+01
	40	3.56E+02±5.42E+01	4.43E+02±4.60E+01	6.66E+02±5.66E+01
Alameda Avenue	20	4.30E+02±4.55E+01	1.20E+03±7.58E+01	1.59E+03±9.50E+01
	30	5.46E+02±5.48E+01	1.14E+03±7.43E+01	1.51E+03±8.59E+01
	40	5.12E+02±5.12E+01	1.23E+03±8.03E+01	1.48E+03±8.51E+01

10.6.3 Scalability Analysis

To study the scalability of our proposal, we now focus on the influence of the two main factors defining the complexity of the instances: the number of traffic logics (20, 30, and 40), and the number of vehicles circulating (100, 300, and 500).

The first observation concerns the number of traffic logics (and hence, the number of traffic lights), since it determines the dimensionality of the problem. In Figure 10.8, we can observe that the mean fitness values increase with the number of traffic logics, as expected. Although, this increment is moderate with regards to the number of signal lights (dotted lines).

A second interesting observation can be obtained from Figure 10.7, where the distribution of results concerning the number of vehicles are completely different for both scenarios. Thus, in Alameda Avenue (three boxplots in the top) the distribution of results gets worse with an increasing number of vehicles. This seems logical since a high number of cars increases the possibility of generating traffic jams. In addition, we must take into account that the number of vehicles that arrive to their destinations directly influences the fitness function. To the contrary, in Rivadavia Square (three boxplots in the bottom) the distributions of results improve as the number of vehicles increases. In this case, we suspect that the particular shape of this scenario, with parallel streets and thus organized flow, could influence in the number of vehicles that quickly reach their destinations and leave the scenario, hence introducing great benefits to the fitness calculation.

10.6.4 Computational Effort

Table 10.5 contains the mean times (and standard deviations) in seconds required by our PSO to compute all the experiments. We must state that these times are averaged, since they were calculated in the scope of a Condor [TTL05] middleware with a pool of machines with different specifications.

The lowest execution time (3.56E+02 seconds) was required for solving the Rivadavia Square scenario with 40 traffic logics and 100 vehicles. The highest time (1.59E+03 seconds) was used in the resolution of Alameda Avenue scenario with 20 signal lights and 500 vehicles. All these times are in a range from 6.33 to 26.5 minutes, which is completely acceptable to the human experts in civil engineering designing and taking decisions on the traffic network.

We stress that the computing time increases with the number of vehicles (common sense), although it decreases with the number of signal lights (counterintuitive). This fact can be due to the optimized cycle programs that control a great number of signal lights. These optimized traffic lights enhance the traffic flow leading the cars to get their destinations quickly, hence reducing the computing load of simulation.

10.6.5 Analysis of Solutions

Finally, in this section we focus on the cycle programs obtained as solutions by our PSO, and the possible profits they can offer to the actual users in this field. Then we show the broad impact of using our approach, able to compute realistic and comprehensive signal light timing programs.

In this context, for each iteration step of the PSO and for each particle in the swarm, we have saved the information obtained from each simulation (solution evaluation) about both, the number of vehicles that reached their destinations and the average duration of their journeys. This way, we can distinguish the progressive improvement in the traffic flow obtained from the initial solutions to the final ones, through out the complete optimization procedure.

A representative example can be observed in the optimization process of the Alameda Avenue scenario with 30 traffic logics (130 signal lights in the cycle program) and 300 vehicles. First, in Figure 10.9 we can see the trace of the number of vehicles that did reach their destinations (upper continuous line) versus the number of vehicles that did not reach their destinations (lower dotted line) for each iteration step in a run of PSO. The overlapped curves show the mean number of vehicles (out of 30 independent runs) that did arrive and did not arrive at their destinations. In contrast, this figure also shows the results (in dotted straight lines) of the SCPG (SUMO algorithm) for this same instance.

We can easily see in Figure 10.9 how the amount of vehicles that did arrive (did not arrive) to their destinations increases (decreases) as the algorithm gets the stop condition of 300 iterations. In fact, at the initial steps of the optimization process, the number of vehicles that reached their destinations was lower than the ones resulting in the cycle program generated by SCPG. However, in the final steps of the PSO procedure, the solutions obtained show a high quality in terms of the traffic flow, since 295 vehicles of the initial 300 (98.33%) finalized their trips successfully. Moreover, a mean number of 255 vehicles completed their trips in the final solutions of PSO (average of 30 runs). This contrast to the 160 vehicles that reached their destinations in the SUMO cycle program. The improvement obtained by our PSO over SCPG is 31.66%.

Another interesting behavior that can be observed in Figure 10.9 is the alternating peaks and valleys that appear in the curves of the single run of PSO. These peaks represent solutions with an accurate fitness but with a low number of cars reaching their destinations. This can be due to the fact that, the fitness function (Equation 10.1) promotes cycle programs with large durations

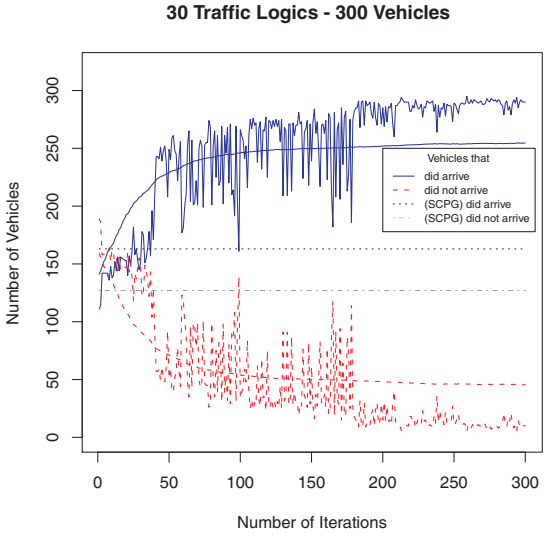


Figure 10.9: Number of Vehicles that did reach their destinations (continuous lines) versus vehicles that did not reach their destinations (dotted lines) in the studied time frame. Overlapped curves show the mean number of vehicles (out of 30 independent runs) that did arrive and did not arrive at their destinations. SCPG results are also shown with dotted straight lines

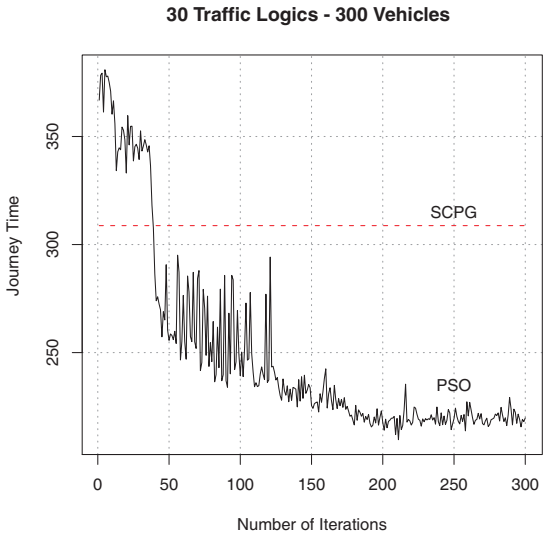


Figure 10.10: Mean trip time of vehicles calculated for each one of the simulations performed through a representative run of PSO. SCPG results are also shown with a dotted straight line. Y axis represents the journey time in seconds

of phases in which the proportion of signal lights in green is higher than in red. For certain intersections with several secondary streets and only one big avenue, the signal lights controlling this avenue could extend their states in red, thus resulting in a traffic jam that could delay the

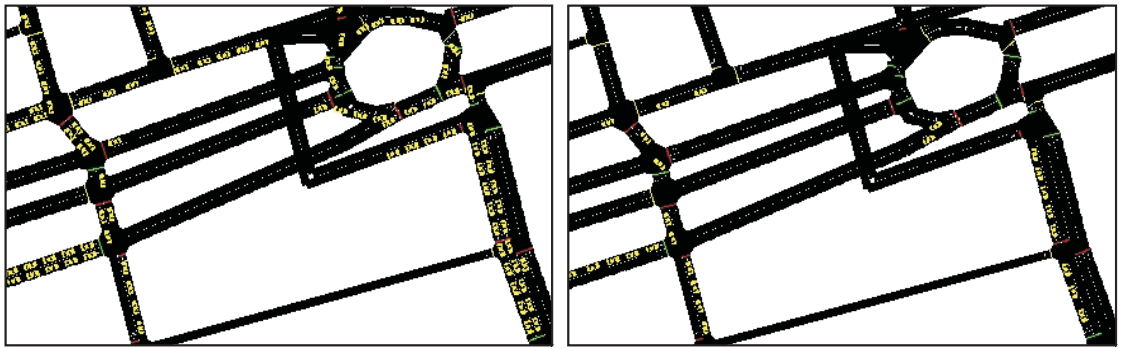


Figure 10.11: Simulation traces of the traffic flow (cars in white) resulting from the cycle programs generated by both, SCPG (left) and PSO (right). The pictures show snapshots at the end of the simulation time. The reader can see that the SCPG leaves a dense traffic while PSO has cleared the routes and the traffic is very fluid and smooth

traffic in other adjacent intersections/streets. A string influence on the successful trips in the fitness function (promoting the number of vehicles that arrive and penalizing when vehicles do not arrive) leads the PSO to avoid this kind of solutions.

From a different point of view, Figure 10.10 plots the trace of the average journey time employed by the vehicles in the resulting solutions of PSO through all the iterations of an example run. In this case, the journey time becomes shorter as the algorithm approaches the stop condition. We must note that, in the calculation of the journey time, the vehicles that did not arrive at their destinations took 500 seconds, the complete simulation time. For this reason, SCPG solutions showed an average journey time of 308.75 seconds while PSO solutions obtained a journey length of 78 seconds, which involves an improvement of 74% with respect to the SCPG solution. In this specific case, 295 vehicles (of 300) completed their trip during the simulation time with an average journey time of 78 seconds to complete the urban scenario of 650 $\times$ 650 meters. In the worst case, the remaining five vehicles will complete their trips in at most 500+78 seconds, that is, the complete simulation time plus the average journey time.

Finally, with the aim of clarifying the final implications of using (or not using) an optimized cycle program, Figure 10.11 shows the simulation traces of the traffic flow resulting from solutions generated by both, SCPG (left) and PSO (right). The pictures were captured at the end of the simulation time (500 seconds), and correspond to two simulation procedures of the scenario instance Alameda Avenue with 40 traffic logics (184 traffic lights) and 500 vehicles. As we can observe, the traffic density of the SCPG cycle program is clearly higher than the one of PSO, even showing the former several intersections with traffic jams. As to the PSO cycle program, all intersections are unblocked at the end of the study.

10.7 Conclusions

In this chapter, we propose an optimization technique based on a Particle Swarm Optimization algorithm that can find successful signal light timing programs. For the evaluation of solutions we use SUMO, a well-known microscopic traffic simulator. For this study we have tested two extensive and heterogeneous metropolitan areas located in Bahía Blanca, and Malaga.

Out of these two scenarios, a total number of 18 different numerical instances have been generated depending on the amount of vehicles circulating and the number of signal lights operating. A series of analyses have been carried out from different points of view: the performance of the optimization technique, the scalability, the computational effort, and the quality of solutions. From these, the following conclusions can be extracted:

1. Our PSO solver performs successfully in the generation of optimized cycle programs for big realistic traffic scenarios. For all the instances, our proposal obtained robust results statistically better than the other two compared algorithms: the SUMO cycle programs generator (SCPG) and a Random Search algorithm (RANDOM).

2. In comparison with the Differential Evolution and Standard PSO 2011 algorithms, our PSO also showed a better performance.

3. In the scope of the scenario instances studied here, we can claim that our PSO scales adequately in terms of the number of traffic lights. Concerning a growing number of vehicles, we have characterized how the scenario topology can influence in the scalability power of our proposal, showing accurate results specially with regular route designs.

4. The complete optimization process required a computational mean time in the range from 6.33 to 26.5 minutes, which is completely acceptable for use by human experts in civil engineering. Furthermore, these values suggest that we can still work with larger scenario instances in future experiments.

5. The final solutions obtained by our PSO can improve the number of vehicles that reach their destinations and the mean journey time, for all the instances. In particular, for the Alameda Avenue instance with 30 traffic logics and 300 vehicles, the improvement obtained is around 31.66% in the number of completed trips and 74% in the journey time, regarding SCPG. All this means a real improvement in city traffic.

For future work, we will be tackling the optimal cycle program with other optimization techniques, and in particular other metaheuristics. We are also interested in using other traffic simulators and creating new larger dimension instances, as close as possible to real scenarios of a whole city.

Part IV

Conclusions and Future Work

Chapter 11

Conclusions and Future Work

11.1 Global Conclusions

Let us start this chapter with a summary of our whole work. This PhD Thesis considers the analysis of the Particle Swarm Optimization, as well as the design and implementation of new proposals based on this algorithm. This work also addresses the resolution of complex real-world optimization problems with PSO in the domains of: DNA Microarrays, VANETs Communication Protocols, and Signal Lights Timing Programs. We have reviewed the concepts of metaheuristics, swarm intelligence, and in concrete, particle swarm optimization. Then, we have put special interest in identifying weaknesses in PSO on different problem's landscape characterizations and scalability conditions. After this, we have proposed advanced design issues with optimized learning procedures, new operators and hybridizations, giving rise to improved versions of PSO. Each different research study considered in this thesis entails a thorough experimentation, with statistical validation and comparisons with the current state of the art.

The main contributions of this thesis are enumerated bellow:

1. **Design and analysis of hybrid PSO with DE**. Taking as a base-line method the PSO, our proposal (called DEPSO) used the differential variation scheme employed in DE for adjusting the velocity of PSO's particles. To test DEPSO, we have performed a thorough experimentation in the context of two Special Sessions of Real Parameter Optimization with different sets of functions: MAEB'09/CEC'05 and GECCO BBOB'09 summing up a total number of 74 different functions with dimension: 2, 3, 5, 10, 20, 30, and 40 variables. On MAEB'09 test set, we showed that hybridizing PSO with DE differential operators provides the first algorithm with search capabilities on rotated and non-separable functions, on which the standard PSO initially showed a limited performance. In addition, DEPSO also showed better performance than canonical DE on a considerable number of functions for dimensions 10 and 30. On BBOB'09 noiseless functions, our proposal obtained an accurate level of coverage rate for separable and weak structured noiseless functions. On BBOB'09 test set noisy functions, DEPSO obtained an accurate level of coverage rate for moderate and severe noise multimodal functions. The fact of using the same parameter setting for all functions (and dimensions), together with the relatively small number of evaluations used, lead us to expect that DEPSO can be easily improved for covering noiseless functions with larger dimensions in other future works.

2. **Scalability analysis of PSO and design of new proposals using a velocity modulation mechanism**. We have designed a new comprehensive velocity modulation mechanism for PSO that, joined with a restarting method, led us to enhance the performance of this algorithm on scalable environments. Our hypothesis is that these two new mechanisms can help the PSO to avoid the early convergence and redirects the particles to promising areas in the search space. After an experimentation phase in the scope of the SOCO'10 benchmark of large scale real-parameter functions, we have tested that our proposal, called RPSO-vm, is scalable, as well as highly competitive with regard to other compared optimizers. In concrete, we can remark that: RPSO-vm outperforms the basic PSO, as well as PSO with each new mechanism separately, for all dimensions. In fact, it is the second best algorithm for all dimensions and statistically similar to the best one in comparison with DE, CHC, and G-CMA-ES, well-known optimizers traditionally used for continuous optimization and showing an excellent performance in other benchmarks (CEC'05, CEC'08, BBOB'09, etc.); our proposal obtained the best results for a number of shifted functions with different properties of modality, separability, and composition. In general, we have tested that RPSO-vm performs relatively better in larger dimensions than in smaller ones.

3. **Design of velocity modulation PSO for multi-objective problems**. We have evaluated six MOPSO algorithms over a set of three well-known benchmark problems by using three different quality indicators. In this context, we have observed that OMOPSO is clearly the most salient of the six compared algorithms. However, the results have also shown that all the algorithms are unable to find accurate Pareto fronts for three multi frontal problems. We have studied this issue and we have proposed the use of a velocity modulation mechanism to enhance the search capability in this sense. The resulting algorithm, namely SMPSO, showed significant improvements when compared with respect to OMOPSO and NSGA-II.

4. **Thorough analysis on the number of informant particles in the learning procedure of PSO**. We have generalized the number of informants that take part in the calculation of new particles in PSO. For this, we have created a new version of Informed PSO, called PSOk, with the possibility of managing any neighborhood size k, from 1 informant to all of them in the swarm. The new proposal has been thoroughly analyzed from the point of view of its evolvability. A series of experiments and comparisons have been carried out in the scope of CEC'05 benchmark of functions. The influence of the number of informants, the problem dimension, and the swarm size have been also analyzed. The following conclusions can be extracted that confirm our initial hypothesis: a number of 6 informants in the neighborhood makes the algorithm to perform with high success in practically all tackled functions. In fact, using few informants (<4) leads the PSOk to show a positive fitness-distance correlation, although evolving solutions with poor fitness values and far from global optima. With more than 10 informants, solutions are again correlated, although concentrating on small non interesting regions of the landscape. Using 6 informants is the best trade-off between fitness-distance and fitness quality. Each PSOk version shows quite similar behavior in our experiments independently of the swarm size, and independently of the problem dimension. This means that PSOk is having additional features making it scalable and resistent to constrained execution (memory-restricted, at least). All this means that, at least for the popular continuous benchmarks, researchers should consider PSO6 instead of the standard PSO with 2 informants as their first choice.

5. **Design of hybrid PSO with Multiple Trajectory Search method for non-separable problems**. Following our previous research line and based on aforementioned results, we

here face an important limitation of PSO up to now: its poor performance on non-separable problems. Our hypothesis is that our proposed PSO6 plus a hybridization for continuous search will outperform the best techniques now. We have hybridized our emerged proposal PSO6 with the modern Multiple Trajectory Search (MTS) local search method to tackle non-separable problems more efficiently. The proposed algorithm, PSO6-Mtsls, has been empirically assessed in the scope of CEC'05 and SOCO'10 benchmarks, summing up a number of 40 real-parameter optimization functions. We can ensure that PSO6-Mtsls statistically outperforms IPSO-Powell and G-CMA-ES, and is better ranked than IACOr-Mtsls and IPSO-Mtsls. These algorithms were catalogued as the most prominent approaches in the literature, which is from now on an old result after our contribution. The local search method Mtsls (LS1) seems to be responsible of the successful performance on non-separable and multimodal functions, whereas the learning procedure of PSO6 takes mostly part in rotated ones.

6. **Design and application of a Parallel Geometric PSO to the gene selection in DNA Microarrays' datasets**. The proposed algorithm, called Parallel Multi-Swarm Optimizer (PMSO), performs an island based distributed topology and uses a Support Vector Machine (SVM) classifier to measure the accuracy of selected subsets of genes. This approach is able to improve the sequential versions in terms of computational effort (Efficiency of 85%). PMSO has been experimentally assessed with different population structures on four well-known cancer datasets, identifying specific genes that our work suggests as significant ones. Concretely, with regard to the Leukemia and Lymphoma datasets, we could confirm that the most frequently PMSO reported genes are also the most relevant genes suggested in the original publications from Biology (Golub et al. and Alizadeh et al., respectively) concerning these Microarrays. Thus, parallelism is confirmed as a powerful tool to enhance the basic search sheet of PSO in real problems.

7. **Application of studied techniques to the optimal parameter tuning communication protocols in VANETs**. We have tackled the optimal File Transfer protocol Configuration (FTC) in vehicular ad hoc networks (VANETs) by means of PSO, as well as other four popular metaheuristic algorithms. For this, we have designed a complex system accounting for a flexible simulation structure targeted for optimizing the transmission time, the number of lost packets, and the amount of data transferred in simulated and also realistic VANET scenarios. The experiments, using *ns-2* (well-known VANET simulator), reveal that all algorithms are capable of efficiently solving the optimum FTC problem. Nevertheless, we have to highlight that PSO performs statistically better than all algorithms in Urban and statistically better than DE and ES in Highway. The scalability analysis shows PSO keeps the best result even for larger instances. From the point of view of the real world utilization, PSO can reduce 19% of the transmission time in Urban and 25.43% in Highway with regards to human experts configuration of CARLINK, while transmitting the same amount of data (1,024 kBytes). The highest effective data rates obtained by PSO (of 300.39 kBytes/s in comparison with 241.5 kBytes/s of human experts) and DE (292.57 kBytes/s) in Urban lead us to advise the final use of swarm intelligent approaches for the automatic tuning of communication protocols in VANETs. This is important for developing new applications between cars in future smart cities like Malaga and others in the world.

8. **Application of swarm intelligent techniques to the Signal Light Timing in road traffic environments**. We have proposed a PSO approach that can find successful signal light timing programs, an important part of any smart mobility system. For the evaluation

of solutions we have used SUMO, a well-known microscopic traffic simulator, deploying for this study two extensive and heterogeneous metropolitan areas located in Bahía Blanca and Malaga. Out of these two scenarios, a total number of 18 different numerical instances have been generated depending on the amount of vehicles circulating and the number of signal lights operating. Our PSO solver performs successfully in the generation of optimized cycle programs for big realistic traffic scenarios. For all the instances, our proposal obtained robust results statistically better than the other two compared algorithms: the SUMO cycle programs generator (SCPG) and a Random Search algorithm (RANDOM). In comparison with DE and the recent Standard PSO 2011 algorithms, our PSO also showed a better performance. In addition, we can claim that our PSO scales adequately in terms of the number of traffic lights. Concerning a growing number of vehicles, we have characterized how the scenario topology can influence in the scalability power of our proposal, showing accurate results specially with regular route designs. The final solutions obtained by our PSO can improve the number of vehicles that reach their destinations and the mean journey time, for all the instances. In particular, for the Alameda Avenue instance in Malaga, with 30 traffic logics and 300 vehicles, the improvement obtained is around 31.66% in the number of completed trips and 74% in the journey time, regarding SCPG (expert's choices). All this means a real improvement in traffic without making major changes in the infrastructures of the city.

9. **Implementation of multiple PSO versions and new proposals for the MALLBA Library**. We have designed and implemented a series of PSO versions following the C++ skeleton structure provided by the MALLBA Library for metaheuristics. Our implementations are currently being used in different research projects. Indeed, a high number of publications referencing it can be checked at `http://neo.lcc.uma.es/software/mallba/project.php`. Furthermore, practically all approaches used in this thesis have been developed using MALLBA. This way, any researcher can easily reproduce the results presented here.

In summary, during this thesis work we have made a wide number of contributions to the field of particle swarm optimization in several ways. From the algorithmic point of view, new advanced versions have been developed and further analyzed attending to different issues. From the point of view of applications, we have tackled several engineering problems belonging to many diverse areas, showing the utility of our proposals for addressing problems that could arise in academy and industry. **As a global conclusion, we can state that our PSO proposals are first class base-line optimizer able of the best performance in modern benchmarking, as well as in present real-world optimization problems**.

11.2 Future Lines of Research

Throughout this PhD Thesis, several open questions concerning the Particle Swarm Optimization have been solved, although new directions for future work have also appeared.

In general, we will investigate on other elemental features and learning procedures of the PSO algorithm. Besides, we plan to perform analytical investigations on new modern benchmarks test suites (BBOB, CEC'13, etc.) with different function characteristics and dimensions. In addition, despite the successful results in solving the three real world applications tackled here, it is a good idea to focus on these problems from different points of view, like using multi-objective, parallel and/or dynamic versions of this algorithm will lead us to explore new opportunities in such domains of application.

In concrete, as future lines of research, we can identify the following main trends.

1. **Adapting PSO to Tackle Non-Separable Problems.** As we observed in previous chapters (mostly in Part II), one of the main issues affecting the PSO when facing complex problems is the interaction between variables. Two variables interact if they cannot be optimized independently to find the global optimum of an objective function. Variable interaction is commonly referred to as non-separability in continuous optimization [LY12], and also known as epistasis or gene interaction in evolutionary computation [Dav90].

   On the one hand, when no interaction between any pair of the decision variables exists, a problem can be solved by optimizing each of the decision variables independently. This is the case of the raw PSO technique, which performs variable by variable updates without any additional mechanism for managing nor detecting interaction between them. On the other hand, the other extreme case is when all of the decision variables interact with each other and all of them should be optimized together. However, most of the real world problems fall in between these two cases, for which, subsets of decision variables interact with each others forming several clusters of interacting variables.

   A way of dealing with this issue is by using cooperative/coevolving models [LY12], by means of which solutions are decomposed into smaller subsets of dependent variables and optimized separately to the rest of solution subsets. In this thesis, we have opted to develop a hybrid PSO method with Multiple Local Search (in Chapter 7), that uses a variable range control mechanism to perform search decisions through iteration steps of the local search. Nevertheless, these approaches require additional structures for managing the non-separability increasing the algorithmic complexity of PSO. A possible new direction is to reformulate the learning equations performing PSO's dynamics to manage sets (clusters) of dependency between variables. The effect of using such a new formulation for particle's movement is therefore a potential topic for future research.

2. **Adaptive Sets of Informant Neighbors.** One of the problems of using neighborhoods of informant particles is that they increase the difficulty of setting up the algorithm. Several decisions like: how many informants and what topology network will constitute the neighborhood, have to be taken. In this thesis work, the number of informants has been empirically investigated, although using an unstructured topology to provide general conclusions on a wide number of benchmark problems. A step beyond in this sense is to design adaptive neighborhoods with different number of informants and topologies depending on the problem structure.

   The rules established in this, as well as in previous studies [AD05], should be used as a starting point for that goal. These rules should be generally based on statistic indicators obtained throughout the search procedure concerning: particle's diversity, adequateness, evolvability, population entropy, probability of falling in local optima, and other linearly (if possible) complex guides; that should lead the algorithm to automatically control the tradeoff between exploration-exploitation for each kind of problem.

3. **Facing Dynamic Problems.** The application of PSO to dynamic problems has been explored by various authors [PV02, HE02, KSA+12]. Similar to EAs, PSO must be modified to successfully performing on dynamic environments with moving peaks of optima. The origin of the difficulty lies in the double problems of outdated memory due to environment dynamism, and diversity loss, due to convergence. From these, diversity loss is critical for

PSO since, the time taken for a partially converged swarm to re-diversify, find the shifted peak, and then re-converge is quite deleterious to its performance [Bla07]. Clearly, either a re-diversification mechanism must be employed at (or before) function change, and/or an indication of diversity can be measured and used throughout the run.

In this sense, we plan to design new mechanisms for either re-diversification or diversity maintenance in PSO based on: randomization, repulsion, dynamic, restarting, and multi-populations; with the aim of facing real word problems characterizing dynamic environments. An example may consists on the optimal Timing Programs of Signal Light on changing road traffic conditions in smartcities.

4. **Reducing Vehicle Emissions and Fuel Consumption.** Nowadays, current cities are continuously increasing levels of pollution emissions and fuel consumption derived from the road traffic directly affect to the air quality, the economy, and specially the health of citizens. Therefore, improving the traffic flow is a mandatory task in order to mitigate such critical problems.

   Following our research study on Timing Programs of Signal Lights, we will investigate on improving the traffic flow of vehicles with the global target of reducing their fuel consumption and gas emissions (CO_2 and NOx). Therefore, using the standard traffic emission model HBEFA [CTS+05] in our algorithmic proposals, we hope to obtain significant reductions in terms of the emission rate and the total fuel consumption, in comparison with timing programs of signal lights predefined by experts close to real ones.

5. **Deploying Swarms in Smart Devices.** We are also interested in formulating new swarm intelligent models capable of evolving as collaborative agents running in communicating portable devices in smart communication networks. The idea is to deploy connecting particles as agents acting in smartphones, tablets, laptops, routers, and other devices, to carry out a collective task, giving rise to modern applications in different areas of interest. In this way, we will implement swarms of agents for corporative company devices collecting information about traffic, software viruses, locations and logistic issues, human flow, commercial preferences, tourist interest points, etc.

All these will lead us create modern applications based on a swarm intelligence design model. In this sense, we will try to directly using real life testing scenarios, with the final aim of assisting the human expert in the decision making process.

Part V

Appendices

Appendix A

Publications Supporting this PhD Thesis Dissertation

In this appendix, we present the set of scientific articles that have been published during the years in which this thesis has been developed. These publications speak for the interest, validity, and impact on the scientific community and literature of the work contained in this thesis, since they have appeared in impact fora, and have been subjected to peer review by expert researchers. Figure A.1 shows a diagram of the different publications and their relationships with the contents of the work.

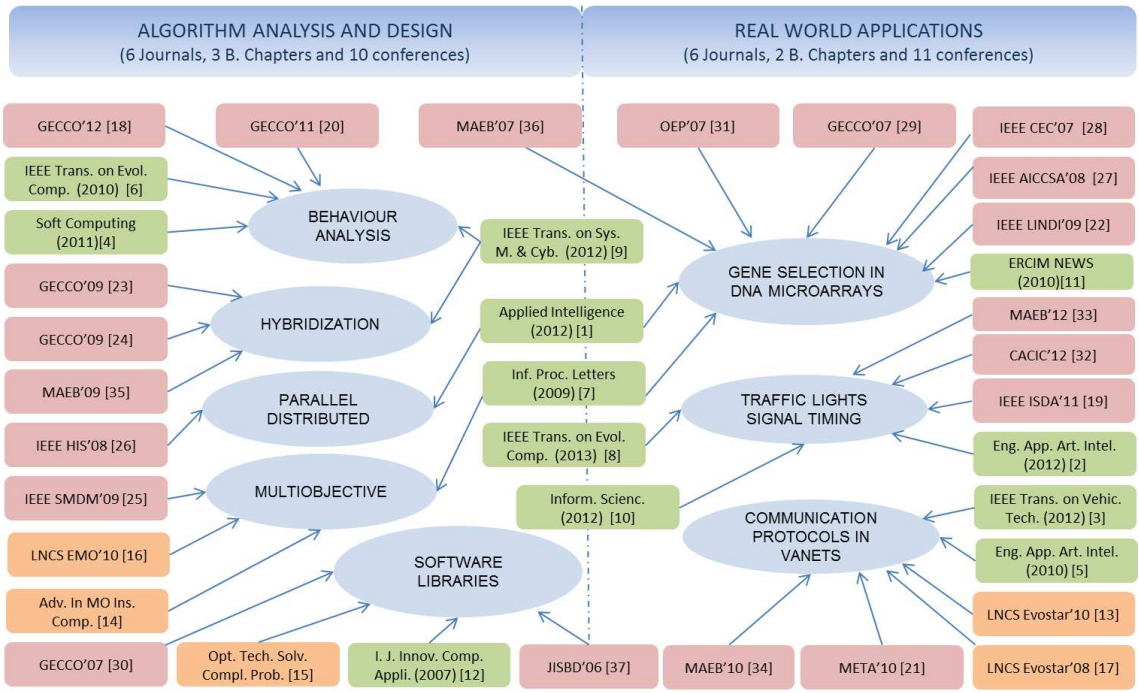


Figure A.1: Diagram of publications generated in the scope of this PhD Thesis

International Journals indexed by ISI-JCR

[1] J. García-Nieto, E. Alba. *Parallel Multi-Swarm Optimizer for Gene Selection in DNA Microarray.* Applied Intelligence, 37(2):255-266, 2012.

[2] J. García-Nieto, E. Alba, A. C. Olivera. *Swarm intelligence for traffic light scheduling: Application to real urban areas.* Engineering Applications of Artificial Intelligence. 25(2):274-283, March 2012.

[3] J. Toutouh, J. García-Nieto, E. Alba. *Intelligent OLSR Routing Protocol Optimization for VANETs.* IEEE Transactions on Vehicular Technology, 60(4):1884-1894, 2012.

[4] J. García-Nieto, E. Alba. *Restart Particle Swarm Optimization with Velocity Modulation:A Scalability Test.* Soft Computing, A Fusion of Foundations, Methodologies and Applications. 15(11):2221-2232. 2011.

[5] J. García-Nieto, J. Toutouh, E. Alba. *Automatic Tuning of Communication Protocols for Vehicular Ad-Hoc Networks Using Metaheuristics.* Engineering Applications of Artificial Intelligence. Special Issue: Advances in metaheuristics for hard optimization: new trends and case studies. 23(5):795-805, August 2010.

[6] J. J. Durillo, A. J. Nebro, C. A. Coello Coello, J. García-Nieto, F. Luna, E. Alba. *A Study of Multi-Objective Metaheuristics when Solving Parameter Scalable Problems.* IEEE Transaction on Evolutionary Computation (TEC), 14 (4):618-635, 2010.

[7] J. García-Nieto, E. Alba, L. Jourdan, E-G. Talbi. *Sensitivity and Specifity Based Multiobjective Approach for Feature Selection: Application to Cancer Diagnosis.* Information Processing Letters, 109(16):887-896. 31 July 2009.

International Journals indexed by ISI-JCR under review

[8] J. García-Nieto, A. C. Olivera, E. Alba. *Optimal Cycle Program of Traffic Lights with Particle Swarm Optimization.* IEEE Transaction of Evolutionary Computation, Submitted, 2013.

[9] J. García-Nieto, E. Alba. *Hybrid PSO6 for Hard Continuous Optimization.* IEEE Transaction on Systems, Man, and Cybernetics-Part B: Cybernetics, Submitted, 2012.

[10] J. García-Nieto, A. C. Olivera, E. Alba. *Reducing Vehicle Emissions and Fuel Consumption in the City by Using Particle Swarm Optimization.* Information Sciences, Submitted, 2012.

International Journals

[11] J. García-Nieto, E. Alba. *Swarm Intelligence Approach for Accurate Gene Selection in DNA Microarrays.* ERCIM-NEW Special Theme: Computational Biology. 82, pp. 26-28, 2010.

[12] E. Alba, G. Luque, J. García-Nieto, G. Ordoñez, G. Leguizamón, *MALLBA: A Software Library To Design Efficient Optimization Algorithms* International Journal of Innovative Computing and Applications (IJICA), 1(1):74 - 85 2007.

Book Chapters and LNCS Series

[13] J. García-Nieto, E. Alba. *Automatic Parameter Tuning with Metaheuristics of the AODV Routing Protocol for Vehicular Ad-Hoc Networks*. In LNCS of the Seventh European Workshop on the Application of Nature-inspired Techniques to Telecommunication Networks and other Connected Systems, (EvoCOMNET'10) EvoWorkshops10, Springer-Verlag, pp. 21-30, Istanbul, 2010

[14] J. J. Durillo, A. J. Nebro, J. García-Nieto, and E. Alba. *On the Velocity Update in Multi-objective Particle Swarm Optimizers*. Advances in multi-objective nature inspired computing, Dhaenens, C., Vermeulen-Jourdan, L., Coello Coello, C.A. (eds.), pp. 45-62. Springer Berlin / Heidelberg, 2009

[15] J. García-Nieto, F. Chicano, E. Alba, *ROS: Remote Optimization Service*. Optimization Techniques for Solving Complex Problems. E. Alba et al. (eds.). pp. 433-457, Wiley & Sons, 2009

[16] J. J. Durillo, J. García-Nieto, A. J. Nebro, C. A. Coello Coello, F. Luna and E. Alba, *Multi-Objective Particle Swarm Optimizers: An Experimental Comparison*. In LNCS 5th International Conference on Evolutionary Multi-Criterion Optimization (EMO'2009), Springer Berlin / Heidelberg, pp. 495-509, Nantes, France, April 7-10, 2009.

[17] E. Alba, J. García-Nieto, J. Taheri, and A. Zomaya. *New Research in Nature Inspired Algorithms for Mobility Management in GSM Networks*. In LNCS of the Fifth European Workshop on the Application of Nature-inspired Techniques to Telecommunication Networks and other Connected Systems, EvoWorkshops08, Springer-Verlag, pp. 1-10, Napoli Italy, 2008.

International Conferences

[18] J. García-Nieto, and E. *Alba.Why Six Informants Is Optimal in PSO*. In ACM Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'12). pp.25-32, Philadelphia, USA, July 2012.

[19] J. García-Nieto, A. C. Olivera, and E. Alba. *Enhancing the Urban Road Traffic with Swarm Intelligence: A Case Study of Cordoba City Downtown*. In IEEE Proceedings of the 11th International Conference on Intelligent Systems Design and Applications (ISDA'2011). pp. 368-373, Córdoba, Spain, November 2011.

[20] J. García-Nieto, and E. Alba. *Empirical Computation of the Quasi-optimal Number of Informants in Particle Swarm Optimization*. In ACM Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'11). pp.147-154. Dublin, Ireland, July 2011.

[21] J. Toutouh, J. García-Nieto, and E. Alba, *Optimal Configuration of OLSR Routing Protocol for VANETs by Means of Differential Evolution*. In 3rd International Conference on Metaheuristics and Nature Inspired Computing (META'2010), October, 2010, D'Jerba (Tunisia). Published on CD.

[22] J. García-Nieto, E. Alba, and J. Apolloni. *Hybrid DE-SVM Approach for Feature Selection: Application to Gene Expression Datasets*. In Proceedings of the 2nd. IEEE International Symposium on Logistics and Industrial Informatics (LINDI 2009) Linz, pp.-1-6 Austria. 10-12 Sep. 2009.

[23] J. García-Nieto, E. Alba, and J. Apolloni. *Noiseless Functions Black-Box Optimization: Evaluation of a Hybrid Particle Swarm with Differential Operators*. In ACM Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'09). pp. 2231-2237. Montreal, Canada, 2009.

[24] J. García-Nieto, E. Alba, and J. Apolloni. P*article Swarm Hybridized with Differential Evolution: Black-Box Optimization Benchmarking for Noisy Functions*. In ACM Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'09). pp. 2343-2349. Montreal, Canada, 2009.

[25] A. J. Nebro, J. J. Durillo, J. García-Nieto, C. A. Coello Coello, F. Luna and E. Alba. *SMPSO: A New PSO-based Metaheuristic for Multi-objective Optimization*. In 2009 IEEE Symposium on Comp.utational Intelligence in Multicriteria Decision-Making, pp. 66-73, IEEE Press, Nashville, Tennessee, USA, March 30 - April 2, 2009

[26] J. Apolloni, G. Leguizamón, J. García-Nieto and E. Alba. *Island Based Distributed Differential Evolution: An Experimental Study on Hybrid Testbeds*. In Proceedings of the IEEE 8th International Conference on Hybrid Intelligent Systems (HIS 2008), IEEE Computer Society, pp. 696-701, Barcelona, Spain, 2008.

[27] E. G. Talbi, L. Jourdan, J. García-Nieto and E. Alba. *Comparison of population based metaheuristics for feature selection: Application to microarray data classification*. In Proceedings of the 6th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA-08), pp. 45-52, Doha, Qatar, 2008.

[28] E. Alba, J. García-Nieto, L. Jourdan and E. G. Talbi. *Gene Selection in Cancer Classification using PSO/SVM and GA/SVM Hybrid Algorithms*. In Proceedings of the IEEE Congress on Evolutionary Computation (CEC'2007), pp. 284-290, Singapore, 27 Sep, 2007.

[29] J. García-Nieto, L. Jourdan, E. Alba and E. G. Talbi. *A Comparison of PSO and GA Approaches for Gene Selection and Classification of Microarray Data*. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO07), London, UK, 2007.

[30] E. Alba, J. García-Nieto, F. Chicano. *Using Metaheuristics Algorithms Via ROS*. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'07), London, UK, 2007.

[31] E. G. Talbi, L. Jourdan, J. García-Nieto and E. Alba. *Slection dattributs de puce ADN par essaim de particules*. In Optimisation par Essaim Particulaire (OEP 2007), Paris, France, 23 April, 2007.

National Conferences

[32] J. García-Nieto, E. Alba, and A. C. Olivera. *Particle swarm optimization aplicado a la programación de los ciclos de semáforos en Bahía Blanca*. XVIII Congreso Argentino de Ciencias de la Computacin (CACIC'12). Bahía Blanca, 8 a 12 de octubre, 2012.

[33] J. García-Nieto, E. Alba, and A. C. *Olivera Planificación de Ciclos en Semáforos con PSO: Casos de Estudio sobre Málaga y Sevilla*. VIII Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB'12), pp. 117-124. Albacete, 8 a 10 de febrero, 2012.

[34] J. Toutouh, J. García-Nieto, E. Alba. *Configuración Optima del Protocolo de Encaminamiento OLSR para VANETs Mediante Evolución Diferencial*. VII Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB'10), Valencia, 7 a 10 de septiembre, 2010.

[35] J. García-Nieto, J. Apolloni, E. Alba and G. Leguizamn. *Algoritmo Basado en Cúmulos de Partículas y Evolución Diferencial para la Resolución de Problemas de Optimización Continua*. VI Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB'09), Málaga, 11 a 13 de Febrero, pp. 433 - 440, 2009.

[36] E. Alba, J. García-Nieto y G. Luque, *Algoritmos Basados en Cúmulos de Partculas para el Análisis de Microarrays de ADN*. En las Actas del V congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB'07), pp. 419-426, Tenerife, febrero, 2007.

[37] E. Alba, J.G. Nieto y F. Chicano, *ROS: Servicio de Optimización Remota*, En J. Riquelme y P. Botella (eds.) Actas de las JISBD, pp. 509-514, Barcelona 2006.

Appendix B

Resumen en Español

B.1 Introducción

Uno de los aspectos más importantes en la investigación en ciencias de la computación consiste en el análisis y diseño de algoritmos de optimización para la resolución de problemas de alta complejidad (catalogados como NP-Duros) de una manera eficiente, tanto en calidad de la solución como en el coste en recursos computacionales. Las metaheurísticas [Gol89, Glo03] son un tipo de algoritmos de optimización que, por lo general, son capaces de obtener soluciones de gran calidad (muchas veces la solución óptima) en un tiempo aceptable. Por tanto, en el campo de la industria, el uso de este tipo de técnicas es cada vez más extendido, contando ya con un gran número de aplicaciones prácticas en multitud de dominios de especialización.

Dentro del contexto de las metaheurísticas, los algoritmos de Inteligencia Colectiva o *Swarm Intelligence (SI)* están siendo objeto de gran interés en los últimos años. Estas técnicas modelan el comportamiento emergente de agentes simples aunque actuando en coletividad, con la intención de desarrollar el proceso de aprendizaje para la resolución de problemas complejos. En particular, los algoritmos de Cúmulos de Partículas o *Particle Swarm Optimization (PSO)* son unos de los más populares dentro de la familia del swarm intelligence, ya que desde su diseño original propuesto por Kennedy y Eberhart en 1995 [KE95], han sido objeto de estudio en multitud de publicaciones y han sido aplicados a un gran número de problemas de optimización, tanto de índole púramente académica, como a problemas reales encontrados en la industria. Sin embargo, a pesar de esta intensa investigación, todavía existen oportunidades de estudio sobre este tópico, ya que en ciertos tipos de problemas, el algoritmo PSO muestra deficiencias susceptibles de mejora. Además, aún podemos encontrar un gran número de aplicaciones de la vida real para las que su rendimiento todavía no ha sido evaluado.

Nuestra motivación en este trabajo de tesis tiene una doble vertiente. Por una parte, estamos interesados en diseñar nuevas propuestas de algoritmos de cúmulos de partículas que resuelvan, o que en la medida de lo posible, sean capaces de mitigar las principales desventajas que presentan estos algoritmos. A tal respecto, hemos utilizado una metodología para el análisis del comportamiento interno de estos algoritmos y para identificar los principales problemas que manifiestan las versiones actuales de PSO. De cara a la evaluación de la efectividad de las nuevas propuestas, hemos llevado a cabo estudios comparativos desde dos puntos de vista: la calidad de la solución y la escalabilidad en términos de dimensionalidad de los problemas a resolver (número de variables de decisión). En este sentido, hemos seguido los procedimientos experimentales específicos de bench-

marks de funciones estándares (CEC'05, SOCO'10, DTLZ, etc.) y hemos comparado nuestras propuestas algorítmicas con aquellas metaheurísticas más exitosas en el estado del arte actual. Por otra parte, tenemos como objetivo también la resolución de problemas complejos presentes en la industria mediante versiones de PSO, para de esta manera determinar la adaptabilidad de este algoritmo a las diferentes representaciones y posibles escenarios de optimización, con tiempo computacional limitado y con el requisito del manejo de grandes cantidades de datos. En concreto, nos hemos enfocado en esta tesis en tres aplicaciones reales de gran complejidad, abarcando campos muy diferenciados de la industria: la Selección de Genes en Microarrays de ADN, la Configuración Óptima de Protocolos de Comunicación para VANETs y la Programación óptima de Ciclos en Semáforos para la gestión del tráfico rodado.

B.2 Organización de la Tesis

Como ya hemos comentado, este trabajo de tesis tiene una doble perspectiva: propuesta algorítmica y dominio de aplicación, lo cual se refleja en su estructura como documento global. Así, esta memoria se ha organiza en cinco partes. En la primera parte presentamos los fundamentos de base para este trabajo, es decir, las metaheurísticas como familia de técnicas avanzadas de resolución, los algoritmos de cúmulos de partículas como técnica principal objeto de estudio y la metodología que hemos empleado para evaluar y validar los resultados numéricos. La segunda parte está dedicada al análisis, diseño y evaluación de nuestras propuestas algorítmicas. En esta parte se realiza además un gran número de comparativas con el estado del arte en el contexto de benchmarks de funciones estándares. La tercera parte trata los problemas reales abordados describiendo: modelos, formulaciones, instancias, escenarios y validación resultados. Además, se realiza la revisión de la literatura relacionada y una recapitulación de las propuestas formuladas por otros autores para estos problemas, o en su caso, para problemas relacionados en el dominio. En la cuarta parte agrupa las principales conclusiones extraídas a lo largo de la tesis y ofrece los comentarios globales y trabajo futuro. Finalmente, la quinta parte contiene apéndices relativos a las publicaciones que han surgido durante el desarrollo de esta tesis, así como este resumen en español.

A continuación, se describen los contenidos de cada capítulo, los cuales pasaán a ser desarrollados más en detalle en las siguientes secciones.

- **Parte I. Motivaciones y Fundamentos**
  - El Capítulo 2 ofrece una introducción sobre los conceptos principales en el campo de la optimización y las metaheurísticas, incluyendo una clasificación de estas técnicas y haciendo especial mención sobre los algoritmos de inteligencia colectiva. En la última sección se describe el procedimiento estadístico de validación de resultados seguido en los experimentos llevados a cabo en cada investigación.
  - En el Capítulo 3 se presenta el algoritmo Particle Swarm Optimization. Se describen las versiones canónica y estándar de PSO, realizando formulaciones de los modelos teóricos y enumerando las versiones de PSO más representativas, así como otras técnicas relacionadas. La última sección se centra en los benchmarks estándares de funciones de optimización utilizadas para la evaluación de las nuevas propuestas y su comparación con otros algoritmos.

- **Parte II. Propuestas Algorítmicas y Validación**
  - El Capítulo 4 trata sobre la hibridación de PSO con operadores diferenciales presentes en DE para proponer el algoritmo DEPSO. En primer lugar se describe la formulación del nuevo algoritmo y directamente se presenta el marco experimental mediante el cual se evalua su funcionamiento. En este marco, se utiliza un conjunto muy numeroso de funciones pertenecientes a los benchmarks: CEC'05 [SHL+05] test suite y las funciones nosisy/noiseless de BBOB'09 [HAFR09b]. Con todo ésto se realiza una serie de análisis respecto a las características de las distintas funciones y comparativas con otros algoritmos constituyentes del estado del arte.
  - El Capítulo 5 presenta el algoritmo Restarting PSO con Modulación de Velocidad (RPSO-vm). Este algoritmo fue diseñado para el abordaje de problemas de optimización continua de gran escala. La validación empírica, en forma de test de escalabilidad se realiza en el marco propuesto en el número especial de la revista Soft Computing, denominado SOCO'10 [HLM10b]. En este capítulo se realiza además un análisis en cuanto al coste computacional requerido por la propuesta.
  - En el Capítulo 6 se hace un recorrido por las versiones de PSO multi-objetivo (MOPSO) más interesantes y se evaluan en el marco de tres familias de problemas de optimización continua. En este contexto se describe nuestra propuesta, que consiste en un algoritmo MOPSO con modulación de velocidad (SMPSO) y se evalua con respecto a aquellas versiones evaluadas anteriormente y con respecto a NSGAII.
  - El Capítulo 7 contiene una de las investigaciones más interesantes realizadas para esta tesis. Comienza mediante un concienzudo análisis del número específico de partículas informandoras que pueden dotar a PSO con un proceso de aprendizaje optimizado. Este análisis se realiza sobre un gran número de funciones de optimización. Tras éste, se realiza un nuevo análisis en tiempo de ejecución desde el punto de vista de la evolvabilidad [GNA12b], con la intención de arrojar luz sobre por qué cierto número de informadores (alrededor de séis) es la mejor opción en la formulación de PSO. Finalmente, se hibrida el algoritmo resultante, PSO6 como método de base, con MTS [TC08] para generar la nueva propuesta algorítmica PSO6-Mtsls, destinada a formar parte del estado del arte actual para un extenso conjunto de funciones problema en los benchmarks CEC'05+SOCO'10.
- **Parte III. Aplicaciones Reales**
  - El Capítulo 8 presenta el problema de la Selección de Genes en Microarrays de ADN, junto con el algoritmo Parallel Multi-Swarm Optimization (PMSO) propuesto para abordar tal problema de manera eficiente. Este algoritmo está basado en la versión binaria Geometric PSO (GPSO), cuyo modelo de optimización posee probadas cualidades para la selección de características. Se analiza la efectividad de la nueva propuesta sobre cuatro conjuntos de datos conocidos en el área, descubriendo nuevos subconjuntos de genes con gran poder clasificador. En este sentido, se realiza con posterioridad una serie de comparativas con el estado del arte en términos de esfuerzo computacional, porcentaje de reducción y ratio de clasificación. Finalmente, se presenta un análisis desde el punto de vista biológico para validar los genes más frecuentemente seleccionados con respecto a los trabajos iniciales en los que fueron catalogados por expertos en el área.
  - El Capítulo 9 aborda la aplicación de PSO a la Configuración Óptima de Protocolos para VANETs. Este capítulo comienza con una revisión de la literatura asociada y con

la formulación del problema en relación al protocolo de comunicación VDTP, objeto de optimización. Tras esto se realizan una serie de análisis y comparativas desde los enfoques de la calidad y servicio de la red y la escalabilidad del problema. Todo esto sobre diferentes modelos de escenario realistas. Las configuraciones de VDTP optimizadas se comparan a su vez con las aplicadas por los expertos en el marco de aplicaciones VANETs reales.

- En el Capítulo 10 se presenta al problema de la Programación Óptima de Ciclos de Semáforos en entornos de tráfico urbano. Tras la revisión de la literatura relacionada y la formulación del problema, se presenta la estrategia de optimización, la cual consta de un PSO para optimización discreta-entera con el simulador de tráfico SUMO. Para la evaluación de soluciones se han generado instancias realistas de las ciudades de Málaga y Bahía Blanca. Se realizan entonces una serie de análisis de rendimiento y comparativas desde el punto de vista de la calidad de la solución y sobre diferentes escalas de escenarios simulados. Los programas de ciclos resultantes se comparan además con los utilizados por los expertos en este área de aplicación.

- **Parte IV. Conclusiones**

  - El Capítulo 11 contiene una revisión global de este trabajo de tesis, agrupando las principales conclusiones obtenidas tras cada desarrollo de investigación. Además, en vista de los resultados obtenidos, se discuten los objetivos de la tesis en cuanto a su grado de consecución. Finalmente, las futuras líneas de investigación a seguir tras esta tesis son también expresadas en este capítulo.

- **Parte V. Apéndices**

  - En los apéndices se organizan, en primer lugar, las publicaciones relacionadas con esta tesis (A), este resumen en español (B), la lista de tablas, la lista de figuras, la lista de pseudocódigos de algoritmos y finalmente, la bibliografía.

B.3 Fundamentos

En esta sección se explican unas breves nociones fundamentales sobre las metaheurísticas y en concreto el algoritmo Particle Swarm Optimization.

B.3.1 Metaheurísticas

Las metaheurísticas son estrategias de alto nivel que combinan distintos métodos para explorar un espacio de búsqueda generado por problema de optimización. Suelen definirse a modo de plantillas que se deben rellenar empleando información específica del problema sobre el cual han de aplicarse (representación de las soluciones, operadores, etc.) y son capaces de abordar problemas cuyos espacios de búsqueda son muy extensos, para los cuales, la utilización de otro tipo de técnicas como las exactas, son innavordables por el coste computacional. Las metaheurísticas pueden clasificarse dentro de dos categorías, según el número de soluciones que manejan de forma simultánea: las basadas en trayectoria, que tienen una única solución y las basadas en población, que manejan un conjunto de soluciones, o población, de forma simultánea. Algunas metaheurísticas conocidas del primer tipo son el recocido simulado (SA), la búsqueda tabú (TS), búsqueda greedy aleatoria adaptativa (GRASP), la búsqueda de vecindario variable (VNS), o la búsqueda local iterada (ILS).

Algunos ejemplos conocidos del segundo tipo son los algoritmos evolutivos (EA), los algoritmos de estimación de distribuciones (EDA), la búsqueda dispersa (SS), la optimización por colonia de hormigas (ACO) y la optimización por cúmulos de partículas (PSO). Esta última metaheurística constituye la base algorítmica objeto de esta tesis.

En cuanto a los problemas seleccionados en esta tesis, hay dos características principales de que deben ser tenidas en cuenta y que justifican el uso de metaheurísticas. La primera es que todos ellos implican una gran complejidad computacional y por lo tanto requieren de muchos recursos para su resolución. La segunda es que, tanto en la Configuración de Protocolos VANETs como en la Programación de Ciclos para Semáforos, el modelo de evaluación de soluciones se basa en resultados de simulaciones a modo de "caja negra". Por tanto, no se dispone de información suficiente para la formulación de heurísticas, ya que en la mayoría de las situaciones, sólo se dispone de los rangos de las variables de decisión y un valor de adecuación para las soluciones. En este escenario, el uso de PSO en particular aporta un valor añadido, ya que este algoritmo muestra una rápida convergencia a soluciones aceptables, siendo esta propiedad especialmente adecuada en la resolución de problemas pesados por el uso de simuladores externos.

B.3.2 Particle Swarm Optimization

El algoritmo de PSO [KE01] es una metaheurística poblacional inspirada en el comportamiento social de las bandadas de pájaros y las agrupaciones de individuos en general. Fue inicialmente diseñado para la resolución problemas de optimización continua. En PSO, cada solución potencial al problema se codifica mediante la *posición* de una partícula y a la población de partículas se le llama *cúmulo* o *enjambre (swarm)*. Para el desarrollo de PSO seguimos la especificación Canónica del mismo [PCG11]. En este algoritmo, cada posición de partícula $\mathbf{x}^i$ se actualiza cada iteración t mediante la Ecuación B.1.

$$\mathbf{x}_i^{t+1} \leftarrow \mathbf{x}_i^t + \mathbf{v}_i^{t+1} \tag{B.1}$$

donde el término $\mathbf{v}_i^{t+1}$ es la velocidad de la partícula, que viene dada por la Ecuación B.2.

$$\mathbf{v}_i^{t+1} \leftarrow \omega \cdot \mathbf{v}_i^t + U[0,\varphi_1] \cdot (\mathbf{p}_i^t - \mathbf{x}_i^t) + U[0,\varphi_2] \cdot (\mathbf{b}_i^t - \mathbf{x}_i^t) \tag{B.2}$$

En esta fórmula, $\mathbf{p}_i^t$ es la mejor solución personal que la partícula i ha encontrado durante su proceso, $\mathbf{b}_i^t$ es la mejor partícula en un vecindario de n partículas (conocida como *el mejor social*) aleatoriamente seleccionado (uniforme) del cúmulo y w es el factor de inercia de la partícula (que controla el balance entre exploración-explotación). Por último, φ_1 y φ_2 son los coeficientes de aceleración, los cuales controlan el efecto relativo de los mejores personal y social de la partícula, mientras que $U[0,\varphi_k]$ es un valor aleatorio uniforme en el intervalo $[0,\varphi_k],\ k \in 1,2$. Este último se genera de nuevo para cada componente en el vector de velocidad y para cada iteración.

Una versión equivalente para el cálculo de la velocidad fue calculada analíticamente en el estudio desarrollado en [CK02], en el cual se utiliza un coeficiente de constricción χ en lugar de la inercia para dar estabilidad a la dinámica del modelo en el momento de la convergencia y evitar así el fenómeno de la oscilación, como se refleja en la Ecuación B.3.

$$\mathbf{v}_i^{t+1} \leftarrow \chi \cdot \left(\mathbf{v}_i^t + U^t[0,\varphi_1] \cdot (\mathbf{p}_i^t - \mathbf{x}_i^t) + U^t[0,\varphi_2] \cdot (\mathbf{b}_i^t - \mathbf{x}_i^t)\right) \tag{B.3}$$

$$\chi \leftarrow \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|},\ \textit{with}\ \varphi \leftarrow \sum_i \varphi_i,\ \textit{and}\ \varphi > 4 \tag{B.4}$$

Algorithm 12 Pseudocódigo de PSO Canónico

1: $S \leftarrow inicializaCumulo(Ss)$
2: **while** $t < MAXIMUM_t$ **do**
3: **for** cada partícula $\mathbf{x}_i^t$ en S) **do**
4: $\mathbf{v}_i^{t+1} \leftarrow$actualizaVelocidad$(\omega, \mathbf{v}_i^t, \mathbf{x}_i^t, \varphi_1, \mathbf{p}_i^t, \varphi_2, \mathbf{b}_i^t)$ //Ecuaciones B.2 y B.4
5: $\mathbf{x}_i^{t+1} \leftarrow$actualizaPosicion$(\mathbf{x}_i^t, \mathbf{v}_i^{t+1})$ //Ecuación B.1
6: $evaluate(\mathbf{x}_i^{t+1})$
7: $p_i^{t+1} \leftarrow$actualizaMejorLocal(p_i^t)
8: **end for**
9: $b^{t+1} \leftarrow$actualizaLeader(b^t)
10: **end while**

El coeficiente de constricción χ se calcula mediante la Ecuación B.4, a partir de los dos coeficientes de aceleración φ_1 y φ_2, siendo la suma de éstos dos coeficientes la que determina qué valor de χ usar. Por lo general se usan valores, $\varphi_1 = \varphi_2 = 2.05$, dando como resultado $\varphi = 4.1$, y $\chi = 0.7298$ [ES00, Tre03]. Con estos valores, el método de constricción propuesto por Clerc [CK02] consigue la convergencia del algoritmo, ya que la amplitud de la oscilación de las partículas decrece a lo largo del proceso de optimización. Este método aporta la ventaja de no tener que utilizar $Vmax$, ni otros valores de restricción de parámetros para prever la explosión en tiempo de búsqueda. No obstante, mediante otros experimentos [ES00] se analizó al mismo tiempo que la opción de dar valores a $Vmax$ como $Xmax$, es decir, el rango dinámico de cada variable por cada dimensión. El resultado en este caso es un PSO sin parámetros específicos al problema, denominado en este estudio como el Canónico PSO.

El pseudocódigo de Algoritmo 12 describe el PSO Canónico. Este algoritmo comienza con la inicialización de las partículas del cúmulo (Línea 1). Los elementos correspondientes de cada posición de partícula (solución) son inicializados con valores aleatorios (por lo general siguiendo una distribución uniforme). Para un número máximo de iteraciones, cada partícula se mueve a través del espacio de búsqueda mediante la actualización de su velocidad y posición (Líneas 4, 5, y 7), se evalúa (Línea 6) y su mejor posición personal se actualiza también (Línea 9). Finalmente, el algoritmo devuelve como resultado la mejor partícula encontrada.

Tradicionalmente, existe en la literatura relacionada un gran número de trabajos en los cuales los algoritmos propuestos se comparan con versiones de PSO llamadas por los autores como "el estándar PSO", aunque en realidad, si se examinan detenidamente, podemos comprobar cómo estas versiones no son siempre la misma ni utilizan los mismos parámetros. Este hecho motivó a un grupo de investigadores en el área, bajo la supervisión de James Kennedy y Maurice Clerc, a ofrecer una versión consensuada y validada de estándar PSO para ser utilizado por los investigadores en sus experimentos [PCG11]. La intención de este estándar no es la de ser el mejor de todas las versiones, si bien pretende ser más una sugerencia de propuesta cercana a la versión original de 1995, aunque actualmente incorpora modificaciones basadas en avances recientes. De este modo, en 2006 apareció el primer estándar de PSO el cual traía pocos cambios con respecto a la versión canónica. Sin embargo, en los siguientes estándares, 2007 y 2011, ya incorporan una serie de mejoras significativas, sobre todo respecto a la invarianza a la rotación de problemas.

Al mismo tiempo en el que se definieron las versiones estándares, sobre todo en la última década, aparecieron un gran número de versiones que incorporaban nuevas formulaciones y mecanismos adicionales con la doble motivación de: mejorar su comportamiento (competitividad) y adaptar PSO a condiciones particulares de cada problema (versatilidad). En este sentido, se han elaborado revisiones del estado del arte y taxonomías [PKB07, SM09] en las que se reúnen más de cien

versiones de PSO, que fueron clasificadas y catalogadas por similitud de propiedades. Entre estas categorías podemos clasificar PSO por: diferentes codificación de soluciones [KE97] [MCP07], hibridación [MdOSVdED11], nuevas formulaciones de velocidad [Ken03] [LQSB06] [ZZLS11] [LYN12], topologías de vecindario y estructura de cúmulo [MKN04] [GNA11a].

B.4 Propuestas Algorítmicas y Estudios Experimentales

En esta sección se presentan los estudios analíticos y propuestas algorítmicas resultantes de esta tesis. Estos estudios consisten fundamentalmente en generación de nuevos híbridos, nuevas formulaciones de dinámicas de movimiento de partículas, tests empíricos con benchmarks estándares de funciones de optimización continua y comparaciones con algoritmos en el top del estado del arte.

B.4.1 Algoritmo Híbrido DEPSO

Se trata de una nueva propuesta algorítmica que utiliza como método base el procedimiento de aprendizaje de PSO, aunque utilizando para el cálculo de la velocidad el esquema de variación diferencial de DE. En este caso, el factor de variación se compone por dos vectores partículas elegidos del cúmulo de manera aleatoria siguiendo el esquema *DE/rand/1* del DE Canónico. De esta forma, para cada posición de partícula del cúmulo $\mathbf{x}_i$, se genera un vector diferencial $(\mathbf{x}^t_{r1} - \mathbf{x}^t_{r2})$, siendo las partículas $\mathbf{x}^t_{r1}$ y $\mathbf{x}^t_{r2}$ seleccionadas aleatóriamente (semilla uniforme) cumpliendo la restricción $i^t \neq r1^t \neq r2^t$. La nueva velocidad $\mathbf{v}^{t+1}_i$ de dicha partícula i se calcula mediante la siguiente ecuación:

$$\mathbf{v}^{t+1}_i \leftarrow \omega \cdot \mathbf{v}^t_i + F \cdot (\mathbf{x}^t_{r1} - \mathbf{x}^t_{r2}) + U^t[0, \varphi_i] \cdot (\mathbf{b}^t_i - \mathbf{x}^t_i) \tag{B.5}$$

La principal diferencia con respecto al PSO original, radica en que se realiza la operación de variación diferencial en lugar de implementar la operación de influencia personal típica de PSO, con el mejor personal de cada partícula $(\mathbf{p}^t_i - \mathbf{x}^t_i)$ expresado en la Ecuación B.2. Tras el cálculo de la velocidad, se realizan los operadores de cruce y selección típicos de todas las versiones estándares de DE, para finalmente generar la nueva posición de partícula $\mathbf{x}^{t+1}_i$.

Para evaluar DEPSO, hemos realizado una extenso estudio experimental en el marco de dos importantes sesiones especiales de optimización continua, con diferentes conjuntos de funciones: MAEB'09/CEC'05 [SHL+05] y GECCO BBOB'09 [HAFR09a] sumando un total de 74 funciones de optimización diferentes con dimensiones: 2, 3, 5, 10, 20, 30 y 40 variables. Respecto al conjunto de funciones de MAEB'09, la principal observación consiste en que el uso de operadores diferenciales provee al PSO con una mejor capacidad de búsqueda en funciones no separables, sobre las cuales, PSO no suele mostrar un buen comportamiento. Además, DEPSO mostró mejores resultados que DE en un amplio número de funciones para dimensiones 10 y 30. Para las funciones sin ruido "noiseless" del benchmark BBOB'09, nuestra propuesta obtuvo un buen comportamiento en las separables y las débilmente estructuradas. En cuanto a las funciones con ruido "noisy" en el mismo benchmark BBOB'09, DEPSO consiguió mejores resultados para las funciones de ruido moderado y las multimodales severas. El hecho de utilizar exactamente la misma parametrización para todas las funciones y dimensiones, junto con el relativo pequeño número de evaluaciones empleadas, nos lleva a sospechar que el comportamiento de DEPSO podría mejorar de manera significativa para funciones ruidosas con mayores dimensiones.

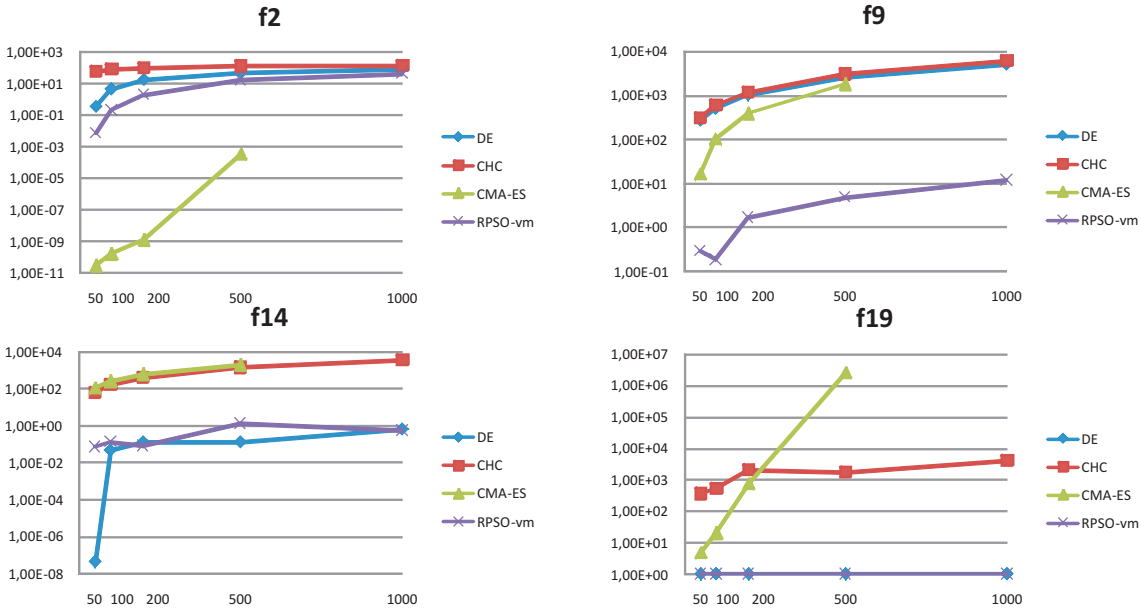


Figure B.1: Resultados en cuanto a la escalabilidad para DE, CHC, G-CMA-ES y RPSO-vm para las funciones f2, f9, f14 y f19. El eje Y muestra los resultados en escala logarítmica. El eje X muestra las dimensiones

B.4.2 PSO con Modulación de Velocidad: Test de Escalabilidad

Uno de los mecanismos que hemos diseñado para PSO en el contexto de esta tesis consiste en la modulación aplicada al vector de velocidad (previamente al cálculo de la nueva posición), que unido a una estrategia de reinicio, nos permite mejorar el comportamiento de este algoritmo cuando nos enfretemos a problemas de gran dimensionalidad. Nuestra hipótesis es que estas dos estratégias pueden ayudar al proceso de búsqueda de PSO a evitar una rápida convergencia y redirigir las partículas hacia áreas más prometedoras en el espacio de búsqueda.

Tras la fase de experimentación, en el contexto del benchmark SOCO'10 [HLM10b] para optimización continua de gran escala, hemos comprobado que nuestra propuesta, a la cual hemos llamado RPSO-vm (Restarting PSO with Velocity Modulation) [GNA11b],es escalable, así como altamente competitiva respecto a algoritmos punteros en el estado del arte. En concreto, podemos puntualizar que RPSO-vm es mejor que PSO, así como mejor que este algoritmo con cada uno de los mecanismos incorporados de manera separada, para todas las dimensiones contempladas (15, 100, 200, 500 y 1000 variables). De hecho, nuestra propuesta resultó ser el segundo mejor algoritmo para todas las dimensiones y estadísticamente similar al mejor de todos en comparación con DE, CHC y G-CMA-ES. Estos tres algoritmos fueron tomados como referencia en el marco experimental recomendado en SOCO'10 ya que constituyen tres potentes optimizadores con buen funcionamiento ya demostrado en otros benchmarks (CEC'05, CEC'08, BBOB'09, etc.); nuestra propuesta además obtuvo los mejores resultados para un gran número de funciones con diferentes propiedades: óptimo desplazado, multimodales, variables no separables y funciones compuestas.

Algorithm 13 Pseudocódigo de MOPSO genérico

1: $S \leftarrow$initializaCúmulo(Ss)
2: $A \leftarrow$initializaArchivoLíderes()
3: calculaCalidadLíderes(A)
4: **while** $t < MAXIMUM_t$ **do**
5: **for** cada partícula i en S **do**
6: $\mathbf{b}^t \leftarrow$ seleccionaLíder(A^t)
7: $\mathbf{v}_i^{t+1} \leftarrow$actualizaVelocidad($\omega, \mathbf{v}_i^t, \mathbf{x}_i^t, \varphi_1, \mathbf{p}_i^t, \varphi_2, \mathbf{b}^t$) //Equations 3.2 o 3.3
8: $\mathbf{x}_i^{t+1} \leftarrow$actualizaPosición($\mathbf{x}_i^t, \mathbf{v}_i^{t+1}$) //Equation 3.1
9: $\mathbf{x}_i^{t+1} \leftarrow$ perturbación($\mathbf{x}_i^{t+1}$)
10: evalía($\mathbf{x}_i^{t+1}$)
11: $\mathbf{p}_i^{t+1} \leftarrow$actualiza($\mathbf{p}_i^t$)
12: **end for**
13: $A^{t+1} \leftarrow$actualizaArchivoLíderes(A^t)
14: calculaCalidadLíderes(A^{t+1})
15: **end while**

Por último, en cuanto a la escalabilidad, la observación más interesante reside en el hecho de que, como muestran las gráficas de Figure B.1 para las funciones f2, f9, f14 y f19 de SOCO'10, RPSO-vm desarrolla una mejor búsqueda para las grandes dimensiones que en las más pequeñas.

B.4.3 PSO con Modulación de Velocidad: Versión Multiobjetivo

Relacionado con el punto anterior, hemos evaluado seis versiones de algoritmos de cúmulo multi-objetivo (MOPSO) sobre tres conjuntos de problemas académicos bien conocidos en el área: ZDT [ZDT00], DTLZ [DTLZ05] y WFG [HHBW06]; utilizando para ello tres indicadores de calidad diferentes: epsilon ($I^1_{\epsilon+}$) [KTZ06] spread (Δ) [DPAM02] e hypervolume (HV) [ZT99].

En el diseño de PSO para adaptarlo al menejo de problemas multi-objetivo hay que tener en cuenta una serie de premisas, siendo la más importante de ellas el hecho de que la solución a un problema con múltiples objetivos es en este caso un conjunto de soluciones no dominadas. Por tanto, se deben tener en cuenta las siguientes consideraicones [RSC06]:

1. Cómo seleccionar las partículas que actuarán como líderes del conjunto de no dominadas.

2. Cómo gestionar las soluciones no dominadas a lo largo del proceso de búsqueda

3. Cómo mantener la diversidad y retrasar la convergencia temprana.

El pseudocódigo de un algoritmo MOPSO genérico se expresa en Algorithm 13. Tras inicializar el swarm (Línea 1), se utiliza un archivo externo para guardar los líderes, los cuales son seleccionadas como las partículas no dominadas en el swarm. Tras inicializar el archivo de líderes (Línea 2), se selecciona alguno de éllos mediante el cálculo de alguna de las medidas de calidad para servir de lider para cada partícula. Ya dentro del bucle principal del algoritmo, las partículas con sus velocidades y posiciones son actualizadas (Líneas 7-8). Opcionalmente se realizará una operación de mutación (factor de *turbulencia*) (Línea 9); entonces, la partícula se evalua y se actualiza su correspondiente *pbest* (Líneas 10-11). Tras cada iteración, el conjunto de líderes es actualizado y las medidas de calidad son calculadas de nuevo (Líneas 13-14). Por último, cuando se alcanza la condición de parada, se devuelve el archivo de líderes como resultado del proceso de búsqueda.

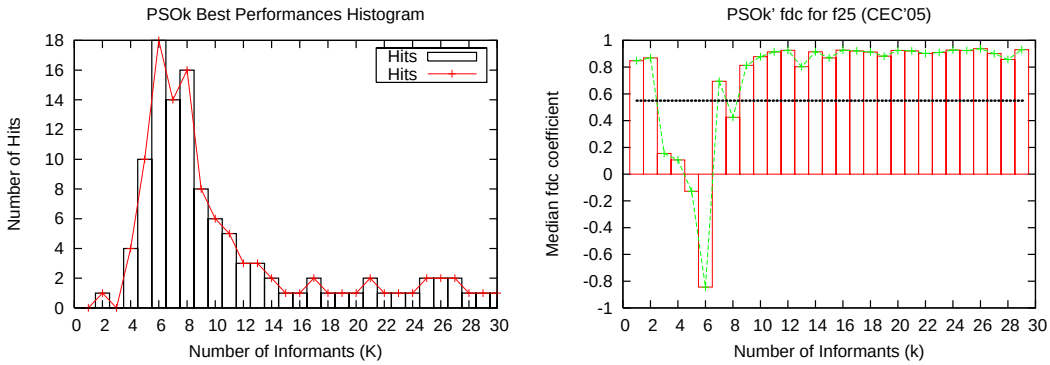


Figure B.2: Histograma de mejores rendimientos (número de Hits). A la derecha se muestran las medianas de coeficientes fdc para as diferentes versiones PSOk, para la función 25 (CEC'05)

En este contexto, hemos observado que OMOPSO [RSC05] es claramente la versión con mejor comportamiento de los algoritmos comparados. Sin embargo, los resultados mostraron además que ninguna versión de PSO multi-objetivo fue capaz de encontrar frentes de Pareto adecuados para tres de los problemas testados, todos ellos de espacio de búsqueda multimodal. No obstante, tras analizar esta desventaja decidimos aplicar el método de modulación de la velocidad para mejorar la capacidad de búsqueda también en estos problemas. El algoritmo resultante, al cual llamamos SMPSO y evaluamos con los mismos conjuntos de problemas, es capaz de obtener mejoras significativas incluso para entornos multimodales, superando incluso el rendimiento de otros algoritmos en el estado del arte como OMOPSO y NSGA-II.

B.4.4 Estudio del Número Óptimo de Informadores en PSO

En el análisis del modelo de aprendizaje que desarrolla PSO, pretendemos en esta tesis arrojar luz sobre el número adecuado de partículas que intervienen como "informantes" para el progreso de cada partícula del cúmulo. De este modo, conseguimos una configuración optimizada de la dinámica de movimiento las partículas, es decir, de su proceso de búsqueda, para un gran número de problemas de optimización con características heterogéneas.

En este sentido, hemos generalizado el número de partículas vecinas informantes que intervienen en el cálculo de nuevas partículas en PSO. Para esta tarea, hemos creado una nueva versión de PSO "*informado*", al cual hemos llamado PSOk, con la posibilidad de manejar cualquier tamaño de vecindario k, desde 1 informador hasta todo el cúmulo como conjunto informante para crear nuevas partículas (del mismo modo que la configuración de FIPS-ALL). La nueva propuesta, con todas sus variantes ha sido analizada mediante la utilización de métricas de evolvabilidad.

Toda la experimentación y comparativas se han realizado en el marco del benchmark de funciones de CEC'05, bajo el cual se analizaron, además de la influencia del número de informantes, la influencia del tamaño de cúmulo y la dimensionalidad de los problemas abordados. Como conclusión principal podemos extraer el hecho de que, efectivamente, un número de 6 partículas informadoras en el vecindario de cada partícula proporciona un mayor rendimiento al algoritmo, para prácticmante todas las funciones de optimización abordadas (véase la Figura B.2). De hecho, por una parte hemos comprobado que utilizar menos informantes (<4) en PSOk lleva al algoritmo a desarrollar una correlación fitness-distancia al óptimo positiva, aunque generando soluciones con

fitness de baja calidad y lejos del óptimo global. Por otra parte, con más de 10 informantes, las soluciones se encuentran también correladas, aunque concentradas en regiones del campo de búsqueda de poco interés. Por tanto, es cuando se utilizan 6 informantes cuando se obtiene el mejor balance entre la calidad del fitness y la distancia al óptimo. Este comportamiento se puede generalizar además independientemente del tamaño del cúmulo, así como la dimensión del problema bordado, dentro del marco de CEC'05.

B.4.5 Nueva Versión: PSO6 con MTS

Basados en los análisis anteriormente presentados, hemos hibridado la propuesta resultante, PSO6, con el reciente método de búsqueda local Multiple Trajectory Search (MTS), para tratar de manera más eficiente los problemas que muestran no separabilidad de variables. Hemos evaluado el algoritmo propuesto, llamado PSO6-Mtsls, en el marco experimental de los benchmarks CEC'05 y SOCO'10, sumando así 40 funciones de optimización continua. Tras los experimentos realizados podemos comprobar como PSO6-Mtsls muestra mejores resultados (validados estadísticamente) que IPSO-Powell y G-CMA-ES, además de obtener un mejor ranking que IACOr-Mtsls y IPSO-Mtsls. Todos estos algoritmos pueden ser catalogados como entre los más prominentes en la literatura relacionada. A modo de conclusión, podemos señalar que el método de búsqueda local Mtsls (LS1) parece ser responsable del mejor comportamiento del algoritmo híbrido en las funciones multimodales no separables, mientras que el procedimiento de aprendizaje que desarrolla PSO6 proporciona un mejor comportamiento en las funciones rotadas.

B.5 Aplicación a Problemas Reales

En esta sección se presentan las propuestas algoritmicas encaminadas a la resolución de problemas reales de gran complejidad: Selección de Genes en Microarrays de ADN, Configuración de Protocolos de Comunicación en VANETs y Programación de Ciclos de Semáforos. Además se incluyen las formulaciones, estratégias de optimización e instancias diseñadas para estos problemas.

B.5.1 Selección de Genes en Microarrays de ADN

Para afrontar el problema de la Selección de Genes en Microarrays de ADN, hemos propuesto un algoritmo, al cual hemos llamando Parallel Multi-Swarm Optimizer (PMSO), que consiste en una serie de Geometric PSOs (GPSO) distribuidos a través de una topología de islas comunicantes en anillo. PMSO utiliza un clasificador de máquinas de vectores de soporte (Support Vector Machine, SVM) para medir la calidad de los subconjuntos de genes seleccionados. El operador de cruce con máscara a partir de tres padres característico de GPSO realiza la labor de selección de genes para la formación de subconjuntos en las soluciones.

Hemos llevado a cabo el estudio experimental de PMSO con diferentes topologías de poblaciones en las islas, para cuatro conjuntos de datos relativos a expresiones de genes en Microarrays bien conocidos por la comunidad científica. En las soluciones resultantes podemos identificar genes recurrentes que nuestro trabajo sugiere como significativos, respecto a todos los demás en cada Microarray. En particular, respecto a los conjuntos de datos Leukemia AML ALL y Lymphoma, podemos confirmar que los genes más frecuentemente seleccionados están dentro de aquellos conjuntos etiquetados como más relevantes en las publicaciones originales de dichos Microarrays (Golub et al. [GST+99] y Alizadeh et al. [Ali00], respectivamente). La Tabla B.1 muestra el conjunto de 11 genes destacados en Golub et al., que también fueron seleccionados por PMSO con mayor

Table B.1: Conjunto de 11 genes destacados en Golub et al., también seleccionados por PMSO con mayor frecuencia para Leukemia

Ranking	Indice	Identificador	Gen Descripción
1	4847	X95735_at	Zyxin
5	1834	M23197_at	CD33 antigen (differentiation antigen)
6	2020	M55150_at	FAH Fumarylacetoacetate
8	3320	U50136_rna1_at	Leukotriene C4 synthase (LTC4S) gene
15	4499	X70297_at	CHRNA7 Cholinergic receptor, nicotinic, alpha polypeptide 7
14	2267	M81933_at	CDC25A Cell division cycle 25A
16	5039	Y12670_at	LEPR Leptin receptor
18	6376	M83652_s_at	PFC Properdin P factor, complement
20	6041	L09209_s_at	APLP2 Amyloid beta (A4) precursor-like protein 2
24	2354	M92287_at	CCND3 Cyclin D3
28	461	D49950_at	Liver mRNA for interferon-gamma inducing factor(IGIF)

frecuencia para Leukemia. Además, en términos de esfuerzo computacional, PMSO es capaz de mejorar la versión secuencial en un 85%, lo cual justifica de por sí el empleo de nuestra técnica distribuida, dada la alta dimensionalidad de los conjuntos de datos al manejar Microarrays y los requerimientos en recursos computacionales.

B.5.2 Configuración Óptima de Protocolos en VANETs

Como segunda aplicación para evaluar el comportamiento de PSO, hemos tratado la Configuración Óptima de Protocolos (File Transfer protocol Configuration, FTC) de comunicación en redes vehiculares ad hoc (VANETs). Debido a que este problema ha sido afrontado por primera vez en esta tesis mediante metaheurísticas, hemos utilizado también otros algoritmos de base: GA, DE, SA, y ES; con el objetivo de establecer comparativas y visualizar la competitividad de PSO en la resolución de este de problema de comunicación real. Por tanto, con este objetivo en mente hemos diseñado e implementado una estructura de optimización (véase la Figura B.3) compuesta por una parte de PSO, así como los algoritmos de optimización evaluados. Por otra parte, se dispone de un simulador de redes VANETs (*ns-2*) encargado de arrojar trazas sobre el tiempo de transmisión, el número de packetes perdidos, la cantidad de datos intercambiados entre los nodos y otros indicadores de QoS de la red, para instancias escenario realistas localizadas en las inmediaciones de la Universidad de Málaga. El simulador acta con el protocolo de comunicación VDTP configurado según los parámetros optimizados en cada solución de PSO.

Los experimentos realizados revelaron que, si bien todos los algoritmos evaluados fueron capaces de dar soluciones eficientes al problema FTC, PSO mostró un rendimiento estadísticamente mejor que las demás técnicas para la instancia Urban, y estadísticamente mejor que DE y ES para la instancia Highway. El análisis de escalabilidad desarrollado mostró que PSO mantiene los mejores resultados para las instancias mayores. Desde el punto de vista de la utilización de las configuraciones resultantes en entornos reales, PSO puede reducir en un 19% el tiempo de transmisión en la instancia Urban y un 25.43% en Highway, respecto a las configuraciones utilizadas por el personal experto del proyecto europeo CARLINK[1]. Como resultado global, podemos destacar que los mayores ratios efectivos de datos enviados obtenidos por PSO (300.39 kBytes/s en comparación con 241.5 kBytes/s de los expertos en el área) y DE (292.57 kBytes/s) en el entorno urbano, nos lleva a recomendar el uso de técnicas de inteligencia colectiva para la configuración automática de protocolos de comunicación en VANETs.

[1]The CARLINK European Project http://carlink.lcc.uma.es

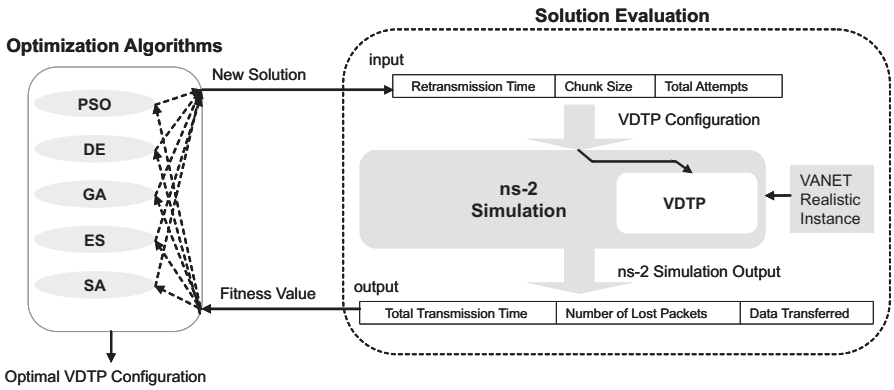


Figure B.3: Estratégia de Optimización del protocolo VDTP para VANETs. Los algoritmos invocan al simulador *ns-2* para cada evaluación de solución

B.5.3 Programación Óptima de Ciclos en Semáforos

El principal objetivo el este problema consiste en encontrar **programas de ciclos optimizados** para todos los semáforos situados en una determinada área urbana. Nos referimos a los programas de ciclos como el periodo de tiempo en el que un conjunto de semáforos (en un cruce) permanecen con sus estados de luces/colores. Al mismo tiempo, estos programas deben también coordinarse con semáforos en intersecciones adyacentes, mejorando así el flujo de vehículos que circulan conforme a la regulación vial establecida.

Para nuestra estrategia de optimización utilizando PSO, hemos codificado los programas de ciclos mediante un vector de números enteros (positivos) siguiendo la estructura del simulador de tráfico SUMO, a través de la cual, cada elemento del vector (variable) representa una duración de fase de los semáforos implicados en una determinada intersección. A pesar de su simplicidad, esta representación permite tener en cuenta la interdependencia entre variables, no sólo para duraciones de fase en una misma intersección, sino también para semáforos en intersecciones adyacentes.

Para evaluar cada programa de ciclos generado por nuestro algoritmo se ha formulado la siguiente función de fitness, mediante la cual se considera la información obtenida de los eventos sucedidos durante la simulación previa:

$$fitness(s) = \frac{\left(\sum_{v=0}^{V} j_v(s)\right) + \left(\sum_{v=0}^{V+C} w_v(s)\right) + (C(s) \cdot St)}{V^2(s) + Cr} \tag{B.6}$$

El objetivo principal (Ecuación B.6) consiste en maximizar el número de vehículos que alcanzan sus destinos (V), así como minimizar el tiempo medio de viaje de todos los vehículos (j_v) durante el tiempo de simulación (St). El número de vehículos que llegan a sus destinos se eleva al cuadrado (V^2) para hacerlo prioritario sobre los demás términos y factores. Obviamente, el número de vehículos que no alcanzan sus destinos y continuan circulando durante la simulación (C) debe ser minimizado. El tiempo medio de viaje se refiere a la agregación de los tiempos de viaje de todos los vehículos que alcanzan sus destinos durante el tiempo de simulación. Por el contrario, los vehículos que no completen sus viajes serán considerados con tiempo de viaje igual al tiempo de simulación, lo cual implica una penalización adicional. Otro importante término considerado en esta función

es el estado en el que los semáforos están en un preciso momento, ya que éste influencia el tiempo que cada vehículo debe parar y esperar (w_v), con el consecuente retardo en su tiempo de viaje.

Finalmente, una proporción bien balanceada de colores en las duraciones de fase de los estados debería promocionar aquellos estados con más semáforos en verde situados en vías o calles con un gran número de vehículos circulando, y los semáforos en rojo situados en calles con un bajo volumen de tráfico. La proporción de colores en casa fase (ph) de todas las intersecciones tl se formula mediante la Ecuación B.7.

$$Cr = \sum_{k=0}^{tl} \sum_{h=0}^{ph} s_{k,h} \cdot \left(\frac{G_{k,h}}{R_{k,h}} \right), \tag{B.7}$$

Donde $G_{k,h}$ es el número de semáforos en verde y $R_{k,h}$ es el número de semáforos en rojo en el estado h (con duración de fase $s_{k,h}$) y en la intersección k. El mínimo valor de $R_{k,h}$ es 1 para evitar la división por cero.

Para la evaluación de las soluciones en forma de programas de ciclos, hemos desarrollado dos instancias basadas en reas metropolitanas reales localizadas en las ciudades de Bahía Blanca en Argentina y Málaga en España. A partir de estos dos escenarios, hemos generado además 18 instancias diferentes dependiendo del número de vehículos circulando (densidad de tráfico) y del número de semáforos en funcionamiento.

Para todas las instancias, nuestra propuesta obtuvo soluciones robustas y mejores estadísticamente que los dos algoritmos comparados inicialmente: SCPG, el generador de ciclos propio de SUMO y RANDOM, un algoritmo de búsqueda aleatoria. Respecto a las otras metaheurísticas comparadas: DE y el Estándar PSO 2011; nuestro PSO también mostró un mejor rendimiento. Además, podemos resaltar que nuestro PSO escala adecuadamente en términos del número de semáforos desplegados. En un entorno creciente de tráfico vehicular, hemos caracterizado también cómo la tipología del escenario puede influenciar a la escalabilidad de nuestra propuesta, mostrando PSO resultados satisfactorios en diseños de rutas regulares, como las cuadrículas desarrolladas en Bahía Blanca. Como producto final, las soluciones optimizadas por nuestro PSO son capaces de mejorar el número de vehículos que llegan a sus destinos en tiempo de simulación, así como el tiempo medio de viaje, para todas las instancias. En particular, para la instancia de Málaga con 30 intersecciones y 300 vehículos, la mejora obtenida ronda el 31.66% en el número de rutas completadas y el 74% en el tiempo de viaje, respecto a SCPG. Todos estos resultados llevan aparejados una mejora real en el tráfico de las ciudades estudiadas.

B.6 Conclusiones

En esta tesis doctoral realizamos un análisis exhaustivo del algoritmo de cúmulos de partículas, y nos centramos en el diseño y la implementación de nuevas propuestas algorítmicas basadas en esta técnica de inteligencia colectiva. Además, tratamos la resolución de problemas complejos reales en los domínios de: los Microarrays de ADN, los Protocolos de Comunicación en redes VANETs y la Programación de Ciclos en semáforos, mediante el uso de PSO. Hemos revisado los conceptos básicos sobre metaheurísticas, inteligencia colectiva y en concreto, particle swarm optimization. En este sentido, menos puesto especial interés en identificar las deficiencias que muestra PSO en diferenes tipologías de problemas y respecto a condiciones de escalabilidad. Tras ésto, hemos propuesto diseños avanzados de mecanismos con procedimientos de aprendizaje optimizados, nuevos operadores y propuestas híbridas, con el objetivo de obtener versiones mejoradas de PSO. Cada

estudio de investigación desarrollado en esta tesis conlleva una completa experimentación, con validación estadística de resultados y comparativas con el estado del arte actual.

Como contribuciones en el campo de PSO realizadas a lo largo de este trabajo de tesis, podemos enumerar las siguientes: desde el punto de vista algorítmico, hemos propuesto operadores de modulación de velocidad, estudiando la escalabilidad y su adaptación a versiones multi-objetivo, hemos desarrollado nuevos híbridos con Evolución Diferencial y Multiple Trajectory Search, y hemos analizado minuciosamente el papel de los vecinos informadores en el proceso de aprendizaje de PSO, resultado el número de séis partículas como de especial interés para la comunidad científica en swarm intelligence. Desde el punto de vista de la aplicación, hemos afrontado tres problemas en diferentes domínios con adaptaciones de PSO para su tratamiento específico, mostrando la utilidad de nuestras propuestas tanto en el ámbito académico, como en el industrial. Como contribución global, podemos declarar que **PSO es un algoritmo de base altamente competitivo, capaz de obtener el mejor rendimiento tanto en benchmarking, como en problemas de optimización reales y de plena actualidad**.

Como futuras líneas de investigación, podemos identificar dos tendencias principales. En primer lugar, estamos centrados en la investigación de otras características elementales del algoritmo PSO, así como en el estudio de otros métodos complementarios para la construcción de nuevas propuestas híbridas, capaces de tratar de manera eficaz las funciones no separables. En este sentido, tenemos planeado realizar análisis de rendimiento mediante la utilización de nuevos benchmarks (BBOB, CEC'13, etc.) con caracterizaciones heterogéneas de funciones y diferentes dimensiones. En segundo lugar, estamos también interesados en la formulación de nuevos modelos de swarm intelligence capaces de funcionar como agentes colaborativos ejecutables en dispositivos inteligentes sobre redes de comunicación. La idea consistiría en desplegar partículas que actuarán como agentes comunicantes en smartphones, tabletas, ordenadores portátiles, routers y otros dispositivos, para llevar a cabo tareas colaborativas, que generen aplicaciones modernas en diferentes áreas de interés. Por ejemplo, cúmulos de partículas en dispositivos corporativos de empresas recolectando información sobre tráfico, virus de software, localizaciones y datos logísticos, flujo humano en grandes superficies, preferencias comerciales, puntos de interés turístico, etc. Todo esto nos llevaría a la creación de aplicaciones modernas basadas en un modelo de swarm intelligence. En este sentido, intentaremos utilizar directamente escenarios reales para la evaluación de nuestras aplicaciones, con la intención final de asistir en la toma de decisiones a los expertos en cada área.

List of Tables

2.1 Different Types of EAs 18

3.1 Parameter settings in Standard PSO 2006 28
3.2 SOCO'10 and CEC'05 benchmark test suites with functions' features 40
3.3 PSO publications by year 41

4.1 Parameter setting used in DEPSO 48
4.2 Error values for functions f_6 to f_{15} with dimension $D = 10$ and 100,000 evaluations 49
4.3 Error values for functions f_{16} to f_{25} with dimension $D = 10$ and 100,000 evaluations 49
4.4 Error values for functions f_6 to f_{15} with dimension $D = 30$ and 300,000 evaluations 50
4.5 Error values for functions f_{16} to f_{25} with dimension $D = 30$ and 300,000 evaluations 50
4.6 Signed Rank test of DEPSO in comparison with G-CMA-ES, K-PCX and DE . . . 51
4.7 Ranking positions of DEPSO for each function with regards to compared algorithms 52
4.8 Noiseless Functions ranked by number of successful trials obtained by DEPSO . . . 53
4.9 Noisy Functions ranked by number of successful trials obtained by DEPSO 53

5.1 Parameter setting used in RPSO-vm 63
5.2 RPSO-vm results for functions f1 to f10 and for all dimensions 64
5.3 RPSO-vm results for functions f11 to f19 and for all dimensions 64
5.4 Mean Errors obtained by RPSO-vm, RPSO, PSO-vm, and PSO for dimension 1,000 65
5.5 Comparison of RPSO-vm versus (lb)RPSO-vm, RPSO, PSO-vm, and PSO 65
5.6 Iman-Davenport's test of RPSO-vm and all compared algorithms 66
5.7 Holm's multicompare test with DE as control 66
5.8 Mean Errors obtained by DE, CHC, G-CMA-ES, and RPSO-vm for dimension 50 . 67
5.9 Mean Errors obtained by DE, CHC, G-CMA-ES, and RPSO-vm for dimension 100 67
5.10 Mean Errors obtained by DE, CHC, G-CMA-ES, and RPSO-vm for dimension 200 68
5.11 Mean Errors obtained by DE, CHC, G-CMA-ES, and RPSO-vm for dimension 500 68
5.12 Mean Errors obtained by DE, CHC, and RPSO-vm for dimension 1000 69
5.13 verage running time (ART) in seconds of the 25 runs of RPSO-vm 71

6.1 Parameter settings 77
6.2 Median and interquartile range of the $I^1_{\epsilon+}$ quality indicator 77
6.3 Median and interquartile range of the Δ quality indicator 78
6.4 Median and interquartile range of the HV quality indicator 79
6.5 Statistical tests between OMOPSO and the rest of the algorithms 79
6.6 NSGA-II vs OMOPSO vs SMPSO: Median and interquartile range of the $I^1_{\epsilon+}$ . . . 81

6.7 NSGA-II vs OMOPSO vs SMPSO: Median and interquartile range of the Δ 82
6.8 NSGA-II vs OMOPSO vs SMPSO: Median and interquartile range of the HV . . . 82
6.9 Statistical tests among NSGA-II, OMOPSO, and SMPSO 83

7.1 Parameter setting used in PSOk 88
7.2 Median error values for the 6 compared algorithms versus PSO6 91
7.3 Average Friedman's Rankings with Holm's correction with PSO-HE{6,8} as control 92
7.4 Complete parameter settings 101
7.5 PSO6: Swarm size parameter tuning 102
7.6 LS1 parameter tuning: best performed values are in bold 102
7.7 Average Friedman's rankings with Holm's correction for SOCO'10 functions 103
7.8 Friedman's rankings with Holm's correction for SOCO'10 featured algorithms . . . 107
7.9 Median Errors of hybridized algorithms with Mtsls 108
7.10 Friedman's rankings with Holm's correction for SOCO'10 and CEC'05 functions . 109
7.11 Number of best median errors with regards to different functions features 110

8.1 Usage details of the four Microarray Datasets 120
8.2 Results of PMSO using 1, 2, 4, and 8 subswarms 123
8.3 Mean time of execution performed by 8-Swarm PMSO 124
8.4 Comparison of PMSO versus PGA, both configured with 8 islands 125
8.5 Comparison of PMSO against six approaches in state of the art 126
8.6 Top 11 genes ranked by Golub et al. also obtained with PMSO 127

9.1 VANET instances specification 135
9.2 Parameter settings applied to PSO and the other techniques 136
9.3 Fitness values with regards to Urban and Highway VANET scenarios 136
9.4 Friedman's and Wilcoxon's Rankings, with PSO as control for VANETs instances . 137
9.5 Mean execution time per independent run of each algorithm 138
9.6 Performance comparison in terms of mean fitness and optimization time 139
9.7 Optimal VDTP configurations and simulation output values 140

10.1 Rivadavia and Alameda instances' specifications 153
10.2 Simulation and PSO parameters 154
10.3 Median fitness values obtained by PSO, RANDOM and SCPG 158
10.4 Median fitness values obtained by PSO, DE and SPSO2011 158
10.5 Mean time and standard deviation in seconds of our PSO 161

B.1 Conjunto de 11 genes destacados en Golub et al. y por PMSO para Leukemia . . . 194

List of Figures

1.1 Conceptual cloud involving the PSO that we have covered in this PhD Thesis . . . 5

2.1 General classification of optimization techniques . . . 12
2.2 Classification of metaheuristics . . . 15
2.3 Structured population models: (left) cellular and (right) distributed . . . 21
2.4 Sorting (*ranking*) of solutions in a bi-objective MOP . . . 22
2.5 Statistical validation procedure for experimental results . . . 24

3.1 Particle swarm optimization vector dynamics . . . 26
3.2 Neighborhood topologies considered in FIPS . . . 33
3.3 Contributions in which any PSO approach is evaluated with an academic benchmark 39
3.4 Summary of PSO publications on real world applications . . . 42

4.1 Median execution trace of DEPSO for dimension 30 . . . 51
4.2 ERT. and N. Suc. Trials of DEPSO on BBOB'09 noiseless functions . . . 54
4.3 ERT. and N. Suc. Trials of DEPSO on BBOB'09 noisy functions . . . 55

5.1 Scaling results for DE, CHC, G-CMA-ES, and RPSO-vm through all dimensions . 70
5.2 Differences in times versus mean error magnitudes of f2 for each dimension . . . 71

6.1 Tracing the velocity of the second variable of OMOPSO when solving ZDT4 . . . 80
6.2 Tracing the velocity of the second variable of SMPSO when solving ZDT4 . . . 80

7.1 neighborhood topologies used by Mendes et al. [MKN04] in FIPS . . . 86
7.2 Performance of the different PSOk versions for the 30 values of k . . . 89
7.3 Mean running time in which all PSOk versions have found the best mean error . . 92
7.4 Influence of the different swarm sizes in PSOk . . . 93
7.5 Influence of different problem dimensions in PSOk . . . 94
7.6 Mean r_{fdc} coefficients of the different PSOk versions . . . 96
7.7 Fitness-distance plots of functions f5, f15, and f24 . . . 97
7.8 Fitness-fitness clouds $\{f', \overline{f(n)}\}$ of functions f5, f15, and f24 . . . 98
7.9 SOCO'10 function's fitness distributions al algorithms for dimension 50 . . . 104
7.10 SOCO'10 function's fitness distribution for dimension 100 . . . 105
7.11 SOCO'10 function's fitness distributions of all featured algorithms . . . 106

8.1 General model of PMSO for gene selection in MAs . . . 118

8.2 Grid-search parameters evaluated in SVM for the datasets 121
8.3 Mean performance of PMSO in different distributions: 1, 2, 4, and 8 subswarms . . 123
8.4 Linear (ideal) speedup versus actual speedup of the 8-Swarm PMSO 125
8.5 Most frequently genes obtained by our 8-S PMSO on Leukemia 128

9.1 Urban VANET scenario with circles representing WiFi coverage 130
9.2 VDTP operation modes: file exchange, timeout expiration, refused communication 132
9.3 Optimization strategy for VDTP configuration in VANETs 133
9.4 Selected area map of Malaga for our VANET instances 134
9.5 Median performance in Urban and Highway scenarios 138
9.6 Three urban areas from Malaga 139
9.7 Effective transmission data rates (kBytes/s) during the simulations 141

10.1 Intersection with four signal lights selected from the SUMO instance map 148
10.2 Optimization strategy for cycle programs of signal lights 151
10.3 Real-world instances for study: Rivadavia Square and Alameda Avenue 152
10.4 Best fitness traces of PSO with different swarm sizes 156
10.5 Best fitness traces of PSO tackling with Rivadavia instance 157
10.6 Swarm fitness histogram through 300 iterations on Rivadavia scenario 157
10.7 Boxplots of distributions results of Rivadavia Square and Alameda Avenue 159
10.8 Increment of the median fitness with regards to the number of signal lights 161
10.9 Vehicles that did reach their destinations versus vehicles that did not 163
10.10Mean trip time of vehicles calculated for simulations 163
10.11Simulation traces of traffic flow resulting from analyzed timing programs 164

A.1 Diagram of publications generated in the scope of this PhD Thesis 177

B.1 Resultados de escalabilidad para DE, CHC, G-CMA-ES y RPSO-vm 190
B.2 Rendimiento de las versiones de PSO-k para $k = \{1 \cdots 30\}$ 192
B.3 Estratégia de Optimización de VDTP para VANETs 195

List of Algorithms

1 Pseudocode of Canonical PSO 27
2 Pseudocode of GPSO for Hamming Spaces 32
3 Pseudocode of Canonical DE 37
4 Pseudocode of DEPSO 47
5 Pseudocode of *Velmod* procedure 60
6 Pseudocode of RPSO-vm 61
7 Pseudocode of a general MOPSO 74
8 Pseudocode of PSO*k* 87
9 Pseudocode of PSO6-Mtsls 100
10 Pseudocode of PMSO 117
11 Pseudocode of RANDOM 155
12 Pseudocódigo de PSO Canónico 188
13 Pseudocódigo de MOPSO genérico 191

Index

Ant Colony Optimization, 13, 19
Artificial Bee Colony, 19

Benchmarking, 37
Benchmarks
    BBOB'09, 38, 45, 52
    CEC'05, 38, 45, 88
    CEC'08, 38
    DTLZ, 73, 76
    SOCO'10, 38, 60, 101
    WFG, 73, 76
    ZDT, 73, 76
    DEJONG'75, 38
    XYAO'99, 38

Classifier
    K-Nearest Neighbor, 116
    Support Vector Machine, 116
Cycle Programs of Signal Lights, 143

Differential Evolution, 36
Diversity, 13
DNA Microarray, 5, 115, 117

Estimation of Distribution Algorithms, 18
Evolutionary Algorithms, 13, 18
Evolvability Analysis, 94
Exploitation, 13
Exploration, 13

Feature Selection Problem, 115
File Transfer Protocol Configuration, 130
Function
    Fitness function, 11
    Objective function, 11

GRASP, 16

Heuristics
    ad hoc, 12
    constructive, 12
    moderns, 13
Hybrid
    DEPSO, 45
    metaheuristic, 20
    PSO6-Mtsls, 99

Intensity, 13
Iterated Local Search, 13, 17

Local optimum, 12
Local search, 12

MALLBA Library, 5
Memetic Algorithm, 20
Metaheuristic, 11
Metaheuristics, 13
    formal definition, 14
    population based, 15, 17
    trajectory based, 15
Mobile Ad Hoc Network (MANET), 131
Multi-objective
    metaheuristic, 22
    optimization, 73
    problem, 22
Multiple Trajectory Search, 17, 99

Neighbourhood
    in a cellular metaheuristic, 21
    in a local search method, 12
NP-hard problem, 3
Ns-2 Network Simulator, 133

Optimization problem
    binary, 11
    continuous, 11
    definition, 11
    heterogeneous, 11
    integer, 11

Optimization techniques
approximate, 12
exact, 11

Parallel models for metaheuristics
cellular, 21
global parallelization, 20
master-slave, 20
structured, 21
Particle Swarm Optimization, 3, 13, 183
Bare Bones, 32
Binary PSO, 30
Canonical, 26
Comprehensive Learning PSO, 33
Discrete PSO, 31
Fylly Informed Particle Swarm, 33, 85
Geometric PSO, 31, 116
Hybrid DEPSO, 45
Incremental Social Learning PSO, 34
Multi-Objective PSO, 6, 73
Orthogonal Learning PSO, 34
Parallel Multi-Swarm Optimizer, 116
PSO6-MTSLS (Multiple Local Search), 99
Standard 2006, 28
Standard 2007, 28
Programming Cycles in Traffic Signals, 143
PSO parameters
Acceleration coeficients, 26
Clerc's constriction coefficient, 26, 187
Inertia Weight, 26

RANDOM, Random Search Algorithm, 154

Scalability Analysis, 66
Scatter Search, 19
SCPG, SUMO Cycle Programs Generator, 155
Signal Lights Timing, 5
Simulated Annealing, 13, 15
Speedup Analysis, 124
Statistical tests, 23
non-parametric, 23
statistical differences, 24
SUMO, Simulator of Urban Mobility, 144, 147
Swarm Intelligence, 3, 15, 183

Tabu Search, 13, 16

Variable Neighbourhood Search, 13, 16
Vehicular Ad Hoc Network (VANET), 5, 129

Bibliography

[ABEF05] J. E. Álvarez-Benítez, R. M. Everson, and J. E. Fieldsend, *A MOPSO Algorithm Based Exclusively on Pareto Dominance Concepts*, EMO 2005 (C. A. Coello Coello et al., ed.), LNCS, no. 3410, 2005, pp. 459–473.

[ABN+99] U. Alon, N. Barkai, D. A. Notterman, K. Gishdagger, S. Ybarradagger, D. Mackdagger, and A. J. Levine, *Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays*, Proceedings of the National Academy of Sciences of the United States of America **96** (1999), no. 12, 6745–6750.

[AD05] E. Alba and B. Dorronsoro, *The exploration/exploitation tradeoff in dynamic cellular genetic algorithms*, IEEE Transactions on Evolutionary Computation **9** (2005), no. 2, 126 – 142.

[AD08] E. Alba and B. Dorronsoro, *Cellular genetic algorithms*, Springer-Verlag, 2008.

[ADL+07] E. Alba, B. Dorronsoro, F. Luna, A.J. Nebro, P. Bouvry, and L. Hogie, *A Cellular Multi-Objective Genetic Algorithm for Optimal Broadcasting Strategy in Metropolitan MANETs*, Computer Communications **30** (2007), no. 4, 685 – 697.

[AGNJT07] E. Alba, J. Garcia-Nieto, L. Jourdan, and E.-G. Talbi, *Gene selection in cancer classification using PSO/SVM and GA/SVM hybrid algorithms*, sep. 2007, pp. 284 –290.

[AH05] A. Auger and N. Hansen, *A restart CMA evolution strategy with increasing population size*, IEEE Congress on Evolutionary Computation **2** (2005), 1769–1776.

[AL05] E. Alba and G. Luque, *Parallel metaheuristics. a new class of algorithms*, Wiley Series on Parallel and Distributed Computing, ch. 2. Measuring the Performance of Parallel Metaheuristics, pp. 43–62, Wiley, 2005.

[Alb02] E. Alba, *Parallel evolutionary algorithms can achieve super-linear performance*, Information Processing Letters, Elsevier **82** (2002), no. 1, 7–13.

[Alb05] _____, *Parallel metaheuristics: A new class of algorithms*, Wiley Series on Parallel and Distributed Computing, Wiley, 2005.

[ALGN+07] E. Alba, G. Luque, J. García-Nieto, G. Ordonez, and G. Leguizamón, *MALLBA: A software library to design efficient optimisation algorithms*, Int. Journal of Innovative Computing and Applications (IJICA) **1** (2007), no. 1, 74–85.

[Ali00] Ash A. Alizadeh, *Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling*, Nature **11** (2000), 403–503.

[ARG+08] E. Angulo, F. P. Romero, R. García, J. Serrano-Guerrero, and J. A. Olivas, *A methodology for the automatic regulation of intersections in real time using soft-computing techniques*, Modelling, Computation and Optimization in Information Systems and Management Sciences, Springer, 2008, pp. 379–388.

[AT01] E. Alba and J. M. Troya, *Analyzing synchronous and asynchronous parallel distributed genetic algorithms*, Future Generation Comp. Sys. **17** (2001), no. 4, 451–465.

[AT02] E. Alba and M. Tomassini, *Parallelism and evolutionary algorithms*, Evolutionary Computation, IEEE Transactions on **6** (2002), no. 5, 443 – 462.

[ATL06] E. Alba, J. Toutouh, and S. Luna, *VDTP: A file transfer protocol for vehicular ad-hoc networks*, Tech. Report D2006/10, University of Malaga, Spain, 2006.

[ATL07] _____, *VanetMobiSim: The vehicular mobility model generator tool for CARLINK*, Tech. Report D1.3.1, University of Malaga, 2007.

[ATL08a] _____, *MEUs ad-hoc communications performance evaluation in highway scenarios (DYNAMIC tests in the CARLINK-UMA scenario)*, Tech. Report D.2.2.5.2008, University of Malaga, Spain, 2008.

[ATL08b] _____, *MEUs ad-hoc communications performance evaluation in urban scenarios (DYNAMIC tests in the CARLINK-UMA scenario)*, Tech. Report D.2.2.4.2008, University of Malaga, Spain, 2008.

[B9̈6] T. Bäck, *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*, Oxford University Press, Oxford, UK, 1996.

[BBSS01] E. Brockfeld, R. Barlovic, A. Schadschneider, and M. Schreckenberg, *Optimizing traffic lights in a cellular automaton model for city traffic*, Phys. Rev. E **64** (2001), no. 5, 056132.

[BHR96] R.D. Bretherton, N.B. Hounsell, and B. Radia, *Public transport priority in scoot*, 1996.

[Bin01] E. Bingham, *Reinforcement learning in neurofuzzy traffic signal control*, European Journal of Operational Research **131** (2001), 232–241.

[Bla07] Tim Blackwell, *Particle swarm optimization in dynamic environments*, Evolutionary Computation in Dynamic and Uncertain Environments (Shengxiang Yang, Yew-Soon Ong, and Yaochu Jin, eds.), Studies in Computational Intelligence, vol. 51, Springer Berlin Heidelberg, 2007, pp. 29–49.

[BR03] C. Blum and A. Roli, *Metaheuristics in combinatorial optimization: Overview and conceptual comparison*, ACM Computing Surveys **35** (2003), no. 3, 268–308.

[car] *Carlink european EUREKA-CELTIC label cp3-005*, [online] Available in URL http://carlink.lcc.uma.es.

[CCAB07] F. Chiang, Z. Chaczko, J. Agbinya, and R. Braun, *Ant-based topology convergence algorithms for resource management in VANETs*, LNCS of EUROCAST, vol. 4739, 2007, pp. 992–1000.

[CCLV07] C. Coello Coello, Gary B. Lamont, and David A. van Veldhuizen, *Evolutionary algorithms for solving multi-objective problems*, Springer Series, 2007.

[CDG05] G.A. Di Caro, F. Ducatelle, and L. M. Gambardella, *AntHocNet: An Adaptive Nature-Inspired Algorithm for Routing in Mobile Ad Hoc Networks*, European Transactions on Telecommunications **16** (2005), no. 5, 443–455.

[Chi07] F. Chicano, *Metaheurísticas e Ingeniería del Software*, Ph.D. thesis, University of Malaga, 2007.

[CK02] M. Clerc and J. Kennedy, *The particle swarm - explosion, stability, and convergence in a multidimensional complex space*, IEEE Transactions on Evolutionary Computation **6** (2002), no. 1, 58 – 73.

[CL02] C-C. Chang and C-J. Lin, *LIBSVM: A library for support vector machines*, 2002.

[Cle99] M. Clerc, *The swarm and the queen: towards a deterministic and adaptive particle swarm optimization*, Congress on Evolutionary Computation CEC'99 (Washington, DC), Piscataway, NJ, 1999, pp. 1951–1957.

[Cle05] ———, *Binary Particle Swarm Optimisers: Toolbox, Derivations, and Mathematical Insights*, 2005.

[Cle10] M. Clerc, *Particle swarm optimization*, Wiley, 2010.

[CMPDDM10] S. Consoli, J. Moreno-Pérez, K. Darby-Dowman, and N. Mladenović, *Discrete particle swarm optimization for the minimum labelling steiner tree problem*, Natural Computing: an international journal **9** (2010), no. 1, 29–46.

[CMRR02] V. D. Cung, S. L. Martins, C. C. Ribeiro, and C. Roucairol, *Strategies for the parallel implementation of metaheuristics*, Essays and Surveys in Metaheuristics, Kluwer Academic Publishers, 2002, pp. 263–308.

[CTS+05] C. Colberg, B. Tona, W. Stahel, M. Meier, and J. Staehelin, *Comparison of a road traffic emission model (HBEFA) with emissions derived from measurements in the Gubrist road tunnel, Switzerland;*, Atmospheric Environment **39** (2005), no. 26, 4703–4714.

[CV95] C. Cortes and V. Vapnik, *Support-vector networks*, Mac. Lear. **20** (1995), no. 3, 273–297.

[CW07] S. B. Cho and H.H. Won, *Cancer classification using ensemble of neural networks with multiple significant gene subsets*, Applied Intelligence **26** (2007), no. 3, 243–250.

[CX06] J. Chen and L. Xu, *Road-junction traffic signal timing optimization by an adaptive particle swarm algorithm*, ICARCV, 2006, pp. 1–7.

[CZZ+10] W. Chen, Henry S. H. Zhang, J., W. Zhong, W. Wu, and Y. Shi, *A novel set-based particle swarm optimization method for discrete optimization problems*, IEEE Transactions on Evolutionary Computation **14** (2010), no. 2, 278–300.

[DAK08] S. Das, A. Abraham, and A. Konar, *Particle swarm optimization and differential evolution algorithms: Technical analysis, applications and hybridization perspectives*, In Advances of Computational Intelligence in Industrial Systems (2008).

[Dav90] Y Davidor, *Epistasis variance: Suitability of a representation to genetic algorithms*, Complex Systems **4** (1990), 369383.

[Dav01] L. Davis, *Handbook of genetic algorithms*, 2001.

[DDBA08] B. Dorronsoro, G. Danoy, P. Bouvry, and E. Alba, *Evaluation of different optimization techniques in the design of ad hoc injection networks*, Workshop on Optimization Issues in Grid and Parallel Computing Environments, part of the HPCS (Nicossia, Cyprus), 2008, pp. 290–296.

[Deb00] K. Deb, *An efficient constraint handling method for genetic algorithms*, Computer Methods in Applied Mechanics and Engineering **186** (2000), no. 2-4, 311–338.

[Deb01] _____, *Multi-objective optimization using evolutionary algorithms*, John Wiley & Sons, 2001.

[DJ75] K. De Jong, *Ean analysis of the behaviour of a class of genetic adaptive systems*, Ph.D. thesis, University of Michigan, USA, 1975.

[DNL+06] J.J. Durillo, A.J. Nebro, F. Luna, B. Dorronsoro, and E. Alba, *jMetal: A Java Framework for Developing Multi-Objective Optimization Metaheuristics*, Tech. Report ITI-2006-10, Departamento de Lenguajes y Ciencias de la Computación, University of Málaga, E.T.S.I. Informática, Campus de Teatinos, December 2006.

[Dor92] M. Dorigo, *Optimization, Learning and Natural Algorithms*, Ph.D. thesis, Politecnico di Milano, Italy, 1992.

[DPAM02] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, *A fast and elitist multiobjective genetic algorithm: NSGA-II*, IEEE Transactions on Evolutionary Computation **6** (2002), no. 2, 182–197.

[DRIE+08] M. Draminski, A. Rada-Iglesias, S. Enroth, C. Wadelius, J. Koronacki, and J. Komorowski, *Monte Carlo feature selection for supervised classification*, Bioinformatics **24** (2008), no. 1, 110–117.

[DS03] I. Dumitrescu and T. Stützle, *Combinations of local search and exact algorithms*, Proceedings of the 2003 international conference on Applications of evolutionary computing (Berlin, Heidelberg), EvoWorkshops'03, Springer-Verlag, 2003, pp. 211–223.

[DS11] S. Das and P. N. Suganthan, *Problem definitions and evaluation criteria for cec 2011 competition on testing evolutionary algorithms on real world optimization problems*, Tech. report, Jadavpur University, and Nanyang Technological University, India and Singapore, Feb 2011.

[DTLZ05] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, *Scalable Test Problems for Evolutionary Multiobjective Optimization*, Evolutionary Multiobjective Optimization. Theoretical Advances and Applications (A. Abraham, L. Jain, and R. Goldberg, eds.), Springer, 2005, pp. 105–145.

[ES00] R. Eberhart and Y. Shi, *Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization*, Proceedings of the IEEE Congress on Evolutionary Computation CEC'00 (La Jolla, CA, USA), vol. 1, 2000, pp. 84–88.

[Esh91] L.J. Eshelman, *The CHC adaptive search algorithm: how to have safe search when engaging in nontraditional genetic recombination*, Proc. of Foundations of Genetic Algorithms Conf (G. Rawlins and M. Kaufmann, eds.), vol. 1, 1991, pp. 265–283.

[FH51] E. Fix and J. L. Hodges, *Nonparametric discrimination: Consistency properties*, Tech. report, 4, US Air Force School of Aviation Medicine, R. Field, TX, 1951.

[FHRA09a] S. Finck, N. Hansen, R. Ros, and A. Auger, *Real-parameter black-box optimization benchmarking 2009: Presentation of the noiseless functions*, Tech. Report 2009/20, Research Center PPE, 2009.

[FHRA09b] ———, *Real-parameter black-box optimization benchmarking 2009: Presentation of the noisy functions*, Tech. Report 2009/21, Research Center PPE, 2009.

[FR95] T. A. Feo and M. G. Resende, *Greedy randomized adaptive search procedures*, Journal of Global Optimization **6** (1995), 109–133.

[GJH+02] G. J. Gordon, R. V. Jensen, L. Hsiao, S. R. Gullans, J. E. Blumenstock, W. G. Ramaswamy, and D. J. Bueno R. Sugarbaker, *Translation of microarray data into clinically relevant cancer diagnostic tests using gene expression ratios in lung cancer and mesothelioma*, Cancer Res **62** (2002), 4963–4967.

[GKS+09] S. Ghosh, D. Kundu, K. Suresh, S. Das, A. Abraham, B.K. Panigrahi, and V. Snasel, *On some properties of the lbest topology in particle swarm optimization*, Hybrid Intelligent Systems, 2009. HIS '09. Ninth International Conference on, vol. 3, 12-14 2009, pp. 370 –375.

[Glo86] F. Glover, *Future paths for integer programming and links to artificial intelligence*, Comput. Oper. Res. **13** (1986), no. 5, 533–549.

[Glo98] ———, *A template for scatter search and path relinking*, Selected Papers from the Third European Conference on Artificial Evolution (London, UK, UK), AE '97, Springer-Verlag, 1998, pp. 3–54.

[Glo03] ———, *Handbook of metaheuristics*, 2003.

[GMLH09] S. García, D. Molina, M. Lozano, and F. Herrera, *A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005*, Journal of Heuristics **15** (2009), no. 6, 617–644.

[GNA10] J. García-Nieto and E. Alba, *Automatic Parameter Tuning with Metaheuristics of the AODV Routing Protocol for Vehicular Ad-Hoc Networks*, Applications of Evolutionary Computation, Lecture Notes in Computer Science, vol. 6025, Springer Berlin Heidelberg, 2010, pp. 21–30.

[GNA11a] J. García-Nieto and E. Alba, *Empirical computation of the quasi-optimal number of informants in particle swarm optimization*, Proceedings of the 13th annual conference on Genetic and evolutionary computation (New York, NY, USA), GECCO '11, ACM, 2011, pp. 147–154.

[GNA11b] J. García-Nieto and E. Alba, *Restart particle swarm optimization with velocity modulation: a scalability test*, Soft Computing **15** (2011), 2221–2232.

[GNA12a] J. García-Nieto and E. Alba, *Parallel multi-swarm optimizer for gene selection in dna microarrays*, Applied Intelligence **37** (2012), 255–266.

[GNA12b] J. García-Nieto and E. Alba, *Why six informants is optimal in pso*, Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference (New York, NY, USA), GECCO '12, ACM, 2012, pp. 25–32.

[GNAA09a] J. García-Nieto, E. Alba, and J. Apolloni, *Noiseless functions black-box optimization: evaluation of a hybrid particle swarm with differential operators*, Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers (New York, NY, USA), GECCO '09, ACM, 2009, pp. 2231–2238.

[GNAA09b] ———, *Particle swarm hybridized with differential evolution: black box optimization benchmarking for noisy functions*, Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers (New York, NY, USA), GECCO '09, ACM, 2009, pp. 2343–2350.

[GNAAL09] J. García-Nieto, J. Apolloni, E. Alba, and G. Leguizamón, *Algoritmo basado en cmulos de partculas y evolucin diferencial para la resolucin de problemas de optimizacin continua*, MAEB 2009 (VI Congreso español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados) (Málaga), 2009, pp. 433–440.

[GNAO12] J. García-Nieto, E. Alba, and A. C. Olivera, *Swarm intelligence for traffic light scheduling: Application to real urban areas*, Engineering Applications of Artificial Intelligence **25** (2012), no. 2, 274 – 283.

[GNTA10] J. García-Nieto, J. Toutouh, and E. Alba, *Automatic tuning of communication protocols for vehicular ad hoc networks using metaheuristics*, Engineering Applications of Artificial Intelligence **23** (2010), no. 5, 795 – 805.

[Gol89] D. E. Goldberg, *Genetic algorithms in search, optimization and machine learning*, 1st ed., Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[GS10] B. P. Gokulan and D. Srinivasan, *Distributed geometric fuzzy multiagent urban traffic signal control*, IEEE Transactions on Intelligent Transportation Systems **11** (2010), no. 3, 714–727.

[GST+99] R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander, *Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring*, Science **286** (1999), 531–537.

[GWBV02] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, *Gene selection for cancer classification using support vector machines*, Machine Learning **46** (2002), no. 1-3, 389–422.

[HAFR09a] N. Hansen, A. Auger, S. Finck, and R. Ros, *Real-parameter black-box optimization benchmarking 2009: Experimental setup*, Tech. Report RR-6828, INRIA, 2009.

[HAFR09b] Nikolaus Hansen, Anne Auger, Steffen Finck, and Raymond Ros, *BBOB'09: Real-Parameter Black-Box Optimization Benchmarking*, Tech. Report RR-6828, INRIA, 2009.

[HB01] D. A. Hensher and K. J. Button, *Handbook of transport systems and traffic control*, Elsevier Science, Amsterdam ; London, 2001.

[HCH09] C. Huang, Y. Chuang, and K. Hu, *Using particle swarm optimization for QoS in ad-hoc multicast*, Eng. Appl. of Artificial Intelligence **In Press** (2009).

[HDH06] B. Huerta, B. Duval, and J. Hao, *A hybrid GA/SVM Approach for Gene Selection and Classification of Microarray Data*, EvoWorkshops (Rothlauf et al, ed.), vol. 3907, Springer, 2006, pp. 34–44.

[HDH07] J. Hernandez, B. Duval, and J-K. Hao, *A genetic embedded approach for gene selection and classification of microarray data*, LNCS of EvoBio (E. Marchiori el al, ed.), 2007, pp. 90–101.

[HE02] X. Hu and R. Eberhart, *Adaptive particle swarm optimization: Detection and response to dynamic systems*, In Proceedings of the IEEE Congress on Evolutionary Computation, CEC2002. IEEE, 2002, pp. 1666–1670.

[HFB07] J. Härri, F. Filali, and C. Bonnet, *Mobility Models for Vehicular Ad Hoc Networks: A Survey and Taxonomy.*

[HFRA09a] N. Hansen, S. Finck, R. Ros, and A. Auger, *Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions*, Tech. Report RR-6829, INRIA, 2009.

[HFRA09b] _____, *Real-parameter black-box optimization benchmarking 2009: Noisy functions definitions*, Tech. Report RR-6869, INRIA, 2009.

[HG90] F. Heppner and U. Grenander, *A stochastic nonlinear model for coordinated bird flocks*, In The Ubiquity of Chaos, Krasner S., ed. (Washington, DC), Springer-Verlag, 1990, pp. 233–238.

[HHBW06] S. Huband, P. Hingston, L. Barone, and L. While, *A Review of Multiobjective Test Problems and a Scalable Test Problem Toolkit*, IEEE Transactions on Evolutionary Computation **10** (2006), no. 5, 477–506.

[HL09a] F. Herrera and M. Lozano, *ISDA'09 Workshop on Evolutionary Algorithms and other Metaheuristics for Continuous Optimization Problems - A Scalability Test*, Tech. report, University of Granada, Pisa, Italy, December 2, 2009.

[HL09b] ———, *Sesión Especial en Metaheurísticas, Algoritmos Evolutivos y Bioinspirados para Problemas de Optimización Continua*, [online] http://decsai.ugr.es/~lozano/AEBs-Continuo/AEBs.htm, Feb 2009.

[HLM10a] F. Herrera, M. Lozano, and D. Molina, *Components and parameters of de, real-coded chc, and g-cmaes*, Tech. report, SCI2S, University of Granada, Spain, Feb 2010.

[HLM10b] ———, *Test suite for the special issue of soft computing on scalability of evolutionary algorithms and other metaheuristics for large scale continuous optimization problems*, Tech. report, SCI2S, University of Granada, Spain, May 2010.

[HMK03] N. Hansen, S. D. Müller, and P. Koumoutsakos, *Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)*, Evol. Comput. **11** (2003), no. 1, 1–18.

[HR04] K. N. Hewage and J. Y. Ruwanpura, *Optimization of traffic signal light timing using simulation*, WSC '04: Proceedings of the 36th conference on Winter simulation, Winter Simulation Conference, 2004, pp. 1428–1436.

[HRM+08] N. Hansen, R. Ros, N. Mauny, M. Schoenauer, and A. Auger, *PSO facing non-separable and ill-conditioned problems*.

[HSL06] V. L. Huang, P. N. Suganthan, and J. J. Liang, *Comprehensive learning particle swarm optimizer for solving multiobjective optimization problems*, Int. J. Intell. Syst. **21** (2006), no. 2, 209–226.

[HTSL07] P. Holm, D. Tomich, J. Sloboden, and C. Lowrance, *Traffic analysis toolbox volume iv: Guidelines for applying corsim microsimulation modeling software*, Tech. report, National Technical Information Service - 5285 Port Royal Road Springfield, VA 22161 USA - Final Report, 2007.

[HW08] M. Haklay and P. Weber, *OpenStreetMap: User-Generated Street Maps*, IEEE Pervasive Computing **7** (2008), 12–18.

[JKCN05] T. Juliusdottir, E. Keedwell, D. Corne, and A. Narayanan, *Two-phase EA/K-NN for feature selection and classification in cancer microarray datasets*, Comp. Int. in Bioinformatics and Computational Biology, 2005, pp. 1–8.

[JMB01] D. B. Johnson, D. A. Maltz, and J. Broch, *DSR: the dynamic source routing protocol for multihop wireless ad hoc networks*, 139–172.

[KB07] D. Karaboga and B. Basturk, *A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm*, J. of Global Optimization **39** (2007), no. 3, 459–471.

[KB09] S. Kachroudi and N. Bhouri, *A multimodal traffic responsive strategy using particle swarm optimization*, Proceedings of the 12th IFAC Symposium on Transportation Systems (Redondo Beach, CA, USA), 2009, pp. 531–537.

[KBM+05] D. Krajzewicz, E. Brockfeld, J. Mikat, J. Ringel, C. Rössel, W. Tuchscheerer, P. Wagner, and R. Wösler, *Simulation of modern Traffic Lights Control Systems using the open source Traffic Simulation SUMO*.

[KBW06] D. Krajzewicz, M. Bonert, and P. Wagner, *The open source traffic simulation package SUMO*, RoboCup 2006 Infrastructure Simulation Competition (2006).

[KD10] C. Karakuzu and O. Demirci, *Fuzzy logic based smart traffic light simulator design and hardware implementation*, Applied Soft Computing **10** (2010), no. 1, 66 – 73.

[KE95] J. Kennedy and R. Eberhart, *Particle Swarm Optimization*, Proc. of the IEEE International Conference on Neural Networks, vol. 4, 1995, pp. 1942–1948.

[KE97] ———, *A Discrete Binary Version of the Particle Swarm Algorithm*, Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, vol. 5, 1997, pp. 4104–4109.

[KE01] James Kennedy and Russell C. Eberhart, *Swarm Intelligence*, Morgan Kaufmann Publishers, San Francisco, California, 2001.

[Ken03] J. Kennedy, *Bare Bones Particle Swarms*, Proceedings of the 2003 IEEE Swarm Intelligence Symposium, 2003. SIS '03 (Washington, DC, USA), 2003, pp. 80–87.

[KGV83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, *Optimization by Simulated Annealing*, Science, Number 4598, 13 May 1983 **220, 4598** (1983), 671–680.

[KJ98] J. Kohavi and G. H. John, *The wrapper approach*, Feature Selection for Knowledge Discovery and Data Mining, 1998, pp. 33–50.

[KL07] J. N. Knight and M. Lunacek, *Reducing the space-time complexity of the CMA-ES*, GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation (New York, NY, USA), ACM, 2007, pp. 658–665.

[KM02] J. Kennedy and R. Mendes, *Population structure and particle swarm performance*, Proceedings of the Congress of Evolutionary Computation CEC'02 (Washington, DC, USA), vol. 2, IEEE Computer Society, 2002, pp. 1671–1676.

[Kra98] S. Krauß, *Microscopic Modeling of Traffic Flow: Investigation of Collision Free Vehicle Dynamics*, Ph.D. thesis, 1998, p. 116.

[KS98] J. Kennedy and W.M. Spears, *Matching algorithms to problems: an experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator*, Proceedings of the IEEE World Congress on Computational Intelligence, may 1998, pp. 78–83.

[KSA+12] M. Khouadjia, B. Sarasola, E. Alba, L. Jourdan, and E. Talbi, *A comparative study between dynamic adapted PSO and VNS for the vehicle routing problem with dynamic requests*, Applied Soft Computing **12** (2012), no. 4, 1426 – 1439.

[KTZ06] J. Knowles, L. Thiele, and E. Zitzler, *A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers*, Tech. Report 214, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, 2006.

[Kut04] M. Kutz, *Handbook of transportation engineering*, McGraw-Hill, 2004.

[LCJM04] B. Liu, Q. Cui, T. Jiang, and S. Ma, *A combinational feature selection and ensemble neural network method for classification of gene expression data*, BMC Bioinformatics **5** (2004), 136–148.

[LH08] S. Lämmer and D. Helbing, *Self-control of traffic lights and vehicle flows in urban road networks*, Journal of Statistical Mechanics: Theory and Experiment **2008** (2008), no. 4, P04019.

[LI02] J. Liu and H. Iba, *Selecting informative genes using a multiobjective evolutionary algorithm*, Proceedings of the IEEE Congress on Evolutionary Computation, CEC '02, vol. 1, May 2002, pp. 297–302.

[Li03] X. Li, *A Non-dominated Sorting Particle Swarm Optimizer for Multiobjective Optimization*, Genetic and Evolutionary Computation - GECCO 2003, LNCS, vol. 2723, 2003, pp. 37–48.

[LKA04] J. Leung, L. Kelly, and J. H. Anderson, *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, CRC Press, Inc., Boca Raton, FL, USA, 2004.

[LKH01] G. Lim, J. J. Kang, and Y. Hong, *The optimization of traffic signal light using artificial intelligence*, FUZZ-IEEE, 2001, pp. 1279–1282.

[LLIB06] J.A. Lozano, P. Larrañaga, P. Inza, and E. Bengoetxea, *Towards a new evolutionary computation. advances in estimation of distribution algorithms*, Springer Verlag, 2006.

[LLY11] G. Lu, J. Li, and X. Yao, *Fitness-probability cloud and a measure of problem hardness for evolutionary algorithms*, EvoCOP'11, LNCS, vol. 6622, Springer, 2011, pp. 108–117.

[LMdOA+11] T. Liao, M. A. Montes de Oca, D. Aydin, T. Stützle, and M. Dorigo, *An incremental ant colony algorithm with local search for continuous optimization*, Proceedings of the 13th annual conference on Genetic and evolutionary computation (New York, NY, USA), GECCO '11, ACM, 2011, pp. 125–132.

[LQSB06] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, *Comprehensive learning particle swarm optimizer for global optimization of multimodal functions*, Evolutionary Computation, IEEE Transactions on **10** (2006), no. 3, 281–295.

[Luq06] G. Luque, *Resolución de Problemas Combinatorios con Aplicación Real en Sistemas Distribuidos*, Ph.D. thesis, University of Malaga, 2006.

[LY12] X. Li and X. Yao, *Cooperatively coevolving particle swarms for large scale optimization*, IEEE Transactions on Evolutionary Computation **16** (2012), no. 2, 210–224.

[LYN12] C. Li, S. Yang, and T. Nguyen, *A self-learning particle swarm optimizer for global optimization problems*, IEEE Transactions on Systems, Man, and Cybernetics, Part B **42** (2012), no. 3, 627–646.

[MBS09] C. L. Müller, B. Baumgartner, and I.F. Sbalzarini, *Particle swarm cma evolution strategy for the optimization of multi-funnel landscapes*, Evolutionary Computation, 2009. CEC '09. IEEE Congress on, may 2009, pp. 2685 –2692.

[MC99] J. Moore and R. Chapman, *Application of particle swarm to multiobjective optimization*, Tech. report, Departament of Computer Science and Software Engineering, Auburn University, 1999.

[MCP07] A. Moraglio, C. Di Chio, and R. Poli, *Geometric Particle Swarm Optimization*, 10th European conference on Genetic Programming (EuroGP 2007), Lecture Notes in Computer Science, vol. 4445, Springer, Abril 2007.

[MdOAS11] M. A. Montes de Oca, D. Aydin, and T. Stützle, *An incremental particle swarm for large-scale continuous optimization problems: an example of tuning-in-the-loop (re)design of optimization algorithms*, Soft Computing **15** (2011), 2233–2255.

[MdOSVdED11] M. A. Montes de Oca, T. Stützle, K. Van den Enden, and M. Dorigo, *Incremental social learning in particle swarms*, Trans. Sys. Man Cyber. Part B **41** (2011), no. 2, 368–384.

[Men04] R. Mendes, *Population Topologies and Their Influence in Particle Swarm Performance*, Ph.D. thesis, Escola de Engenharia, Universidade do Minho, Portugal, 2004.

[Mer04] P. Merz, *Advanced fitness landscape analysis and the performance of memetic algorithms*, Evol. Comput. **12** (2004), 303–325.

[MH97] N. Mladenovic and P. Hansen, *Variable neighborhood search*, Computers And Operations Research **24** (1997), no. 11, 1097–1100.

[MKN04] R. Mendes, J. Kennedy, and J. Neves, *The Fully Informed Particle Swarm: Simpler, Maybe Better*, IEEE Transactions on Evolutionary Computation **8** (2004), no. 3, 204 – 210.

[MLTPn09] S. Muelas, A. La Torre, and J. Peña, *A memetic differential evolution algorithm for continuous optimization*, Proceedings of the 2009 Ninth International Conference on Intelligent Systems Design and Applications (Washington, DC, USA), ISDA '09, IEEE Computer Society, 2009, pp. 1080–1084.

[MM10] J. McCrea and S. Moutari, *A hybrid macroscopic-based model for traffic flow in road networks*, European Journal of Operational Research **207** (2010), no. 1, 676–684.

[MMWP05] A. S. Mohais, R. Mendes, C. Ward, and C. Posthoff, *Neighborhood re-structuring in particle swarm optimization*, LNCS 3809. Proceedings of the 18th Australian Joint Conference on Artificial Intelligence, Springer, 2005, pp. 776–785.

[MPnLR10] S. Muelas, J. Peña, A. LaTorre, and V. Robles, *A new initialization procedure for the distributed estimation of distribution algorithms*, Soft Computing **15** (2010), no. 4, 713–720.

[MS11] C. L. Müller and I. F. Sbalzarini, *Global characterization of the cec 2005 fitness landscapes using fitness-distance analysis*, Proceedings of the 2011 international conference on Applications of evolutionary computation - Volume Part I (Berlin, Heidelberg), EvoApplications'11, Springer-Verlag, 2011, pp. 294–303.

[MT03] S. Mostaghim and J. Teich, *Strategies for finding good local guides in multi-objective particle swarm optimization (MOPSO)*, Proceedings of the IEEE Swarm Intelligence Symposium, 2003. SIS '03, 2003, pp. 26–33.

[Nag10] T. Nagatani, *Effect of speed fluctuation on green-light path in 2d traffic network controlled by signals*, Physica A: Statistical Mechanics and its Applications **389** (2010), no. 19, 4105–4115.

[NCM11] F. Neri, C. Cotta, and P. Moscato, *Handbook of memetic algorithms*, Studies in Computational Intelligence, Springer, 2011.

[NF77] M. Narendra and K. Fukunaga, *A branch and bound algorithm for feature subset selection*, IEEE Trans. on Computer **26** (1977), 917–922.

[ns2] *The Network Simulator Project - Ns-2*, [online] Available in URL http://www.isi.edu/nsnam/ns/.

[PCG11] PSO-Central-Group, *Standard PSO 2006, 2007, and 2011*, Tech. Report [online] http://www.particleswarm.info/, Particle Swarm Central, January 2011.

[PFE05] G. Pampara, N. Franken, and A.P. Engelbrecht, *Combining particle swarm optimisation with angle modulation to solve binary problems*, Evolutionary Computation, 2005. The 2005 IEEE Congress on, vol. 1, sept. 2005, pp. 89–96.

[PKB07] R. Poli, J. Kennedy, and T. Blackwell, *Particle swarm optimization*, Swarm Intelligence **1** (2007), 33–57, 10.1007/s11721-007-0002-0.

[Pow64] M. J. D. Powell, *An efficient method for finding the minimum of a function of several variables without calculating derivatives*, The Computer Journal **7** (1964), no. 2, 155–162.

[PSL05] K. V. Price, R. Storn, and J. Lampinen, *Differential evolution: A practical approach to global optimization*, Springer-Verlag, London, UK, 2005.

[PSS+94] A.C. Pease, D. Solas, E. Sullivan, M. Cronin, C. P. Holmes, and S. Fodor, *Light-generated oligonucleotide arrays for rapid dna sequence analysis*, Proc. Natl. Acad. Sci. (USA), vol. 96, 1994, pp. 5022–5026.

[PV02] K. Parsopoulos and M. Vrahatis, *Recent approaches to global optimization problems through particle swarm optimization*, no. 2-3, 235–306.

[PWDL09] L. Peng, M.-H. Wang, J.-P. Du, and G. Luo, *Isolation niches particle swarm optimization applied to traffic lights controlling*, dec. 2009, pp. 3318 –3322.

[Rai06] G. R. Raidl, *A unified view on hybrid metaheuristics*, Proceedings of the Third international conference on Hybrid Metaheuristics (Berlin, Heidelberg), HM'06, Springer-Verlag, 2006, pp. 1–12.

[RB06] T. J. Richer and T. M. Blackwell, *The lévy particle swarm*, Evolutionary Computation, 2006. CEC 2006. IEEE Congress on, 2006, pp. 808–815.

[Ree93] C. R. Reeves (ed.), *Modern heuristic techniques for combinatorial problems*, John Wiley & Sons, Inc., New York, NY, USA, 1993.

[Rey87] C. W. Reynolds, *Flocks, herds and schools: A distributed behavioral model*, Proceedings of the 14th annual conference on Computer graphics and interactive techniques (New York, NY, USA), SIGGRAPH '87, ACM, 1987, pp. 25–34.

[RKP05] J. Ronkkonen, S. Kukkonen, and K.V. Price, *Real-parameter optimization with differential evolution*, IEEE Congress on Evolutionary Computation **1** (2005), 506–513.

[RPS00] N. M. Rouphail, B. B. Park, and J. Sacks, *Direct signal timing optimization: Strategy development and results*, Tech. report, In XI Pan American Conference in Traffic and Transportation Engineering, 2000.

[RR09] S. M. Rahman and N. T. Ratrout, *Review of the fuzzy logic based approach in traffic signal control: Prospects in saudi arabia*, Journal of Transportation Systems Engineering and Information Technology **9** (2009), no. 5, 58 – 70.

[RSA10] L. B. Romdhane, H. Shili, and B. Ayeb, *Mining microarray gene expression data with unsupervised possibilistic clustering and proximity graphs*, Applied Intelligence **33** (2010), no. 2, 220–231.

[RSC05] M. Reyes-Sierra and C. Coello, *Improving PSO-Based Multi-objective Optimization Using Crowding, Mutation and ϵ-Dominance*, Evolutionary Multi-Criterion Optimization (EMO 2005), LNCS 3410, 2005, pp. 505–519.

[RSC06] ———, *Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art*, International Journal of Computational Intelligence Research **2** (2006), no. 3, 287–308.

[SA12] C. Salto and E. Alba, *Designing heterogeneous distributed GAs by efficiently self-adapting the migration period*, Applied Intelligence **36** (2012), 800–808, 10.1007/s10489-011-0297-9.

[SC97] J. C. Spall and D. C. Chin, *Traffic-responsive signal timing for system-wide traffic control*, Transportation Research Part C: Emerging Technology **5** (1997), no. 3-4, 153 – 163.

[SCC06] D. Srinivasan, M. C. Choy, and R. L. Cheu, *Neural networks for real-time traffic signal control*, IEEE Transactions on Intelligent Transportation Systems **7** (2006), no. 3, 261–272.

[SGR05] J. Sánchez, M. Galán, and E. Rubio, *Stochastic vs deterministic traffic simulator. comparative study for its use within a traffic light cycles optimization architecture*, Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach (José Mira and José Álvarez, eds.), Lecture Notes in Computer Science, vol. 3562, Springer Berlin / Heidelberg, 2005, pp. 622–631.

[SGR08] ———, *Applying a traffic lights evolutionary optimization technique to a real case: "Las Ramblas" area in Santa Cruz de Tenerife*, Evolutionary Computation, IEEE Transactions on **12** (2008), no. 1, 25–40.

[She07] D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*, Chapman & Hall/CRC, 2007.

[SHL+05] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari, *Problem Definitions and Evaluation Criteria for the CEC'05 Special Session on Real-Parameter Optimization*, Tech. Report KanGAL Report 2005005, Nanyang Technological University, Singapore and Kanpur, India, 2005.

[SM09] D. Sedighizadeh and E. Masehian, *Particle swarm optimization methods, taxonomy and applications*, Int. Jour. of Computer Theory and Engineering **1** (2009), no. 5, 486–502.

[SP95] R. Storn and K. V. Price, *Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces*, Tech. report, TR-95012-ICSI, 1995.

[SQ06] Y.-W. Shang and Y.-H. Qiu, *A note on the extended rosenbrock function*, Evolutionary Computation **14** (2006), no. 1, 119–126.

[STD05] A. Sinha, S. Tiwari, and K. Deb, *A population-based, steady-state procedure for real-parameter optimization*, IEEE Congress on Evolutionary Computation **1** (2005), 514–521.

[SWLH06] A. M. Sutton, D. Whitley, M. Lunacek, and A. Howe, *Pso and multi-funnel landscapes: how cooperation might limit exploration*, Proceedings of the 8th annual conference on Genetic and evolutionary computation (New York, NY, USA), GECCO '06, ACM, 2006, pp. 75–82.

[TAYH09] A. M. Turky, M. S. Ahmad, M. Z. Yusoff, and B. T. Hammad, *Using genetic algorithm for traffic light control system with a pedestrian crossing*, RSKT '09: Proceedings of the 4th International Conference on Rough Sets and Knowledge Technology, 2009, pp. 512–519.

[TC04] G. Toscano and C. Coello, *Using Clustering Techniques to Improve the Performance of a Multi-objective Particle Swarm Optimizer*, Genetic and Evolutionary Computation - GECCO 2004, LNCS, vol. 3102/2004, 2004, pp. 225–237.

[TC08] L. Tseng and C. Chen, *Multiple trajectory search for Large Scale Global Optimization*, IEEE Congress on Evolutionary Computation, 2008, pp. 3052–3059.

[TC09] ———, *Multiple trajectory search for unconstrained/constrained multi-objective optimization*, Proceedings of the Eleventh conference on Congress on Evolutionary Computation (Piscataway, NJ, USA), CEC'09, IEEE Press, 2009, pp. 1951–1958.

[TGNA12] J. Toutouh, J. García-Nieto, and E. Alba, *Intelligent OLSR Routing Protocol Optimization for VANETs*, Vehicular Technology, IEEE Transactions on **61** (2012), no. 4, 1884–1894.

[TLS+10] K. Tang, Xiaodong Li, P. N. Suganthan, Z. Yang, and T. Weise, *Benchmark Functions for the CEC'10 Special Session and Competition on Large Scale Global Optimization*, Tech. Report Nature Inspired Computation and Applications Laboratory, USTC, Nanyang Technological University, anyang Technological University, China, 2010.

[TLTM05] C. Tolba, D. Lefebvre, P. Thomas, and A. El Moudni, *Continuous and timed petri nets for the macroscopic and microscopic traffic flow modelling*, Simulation Modelling Practice and Theory **13** (2005), no. 5, 407 – 436.

[Tre03] I. C. Trelea, *The particle swarm optimization algorithm: convergence analysis and parameter selection*, Information Processing Letters **85** (2003), 317–325.

[TSW07] F. Teklu, A. Sumalee, and D. Watling, *A genetic algorithm approach for optimizing traffic control signals considering routing*, Computer-Aided Civil and Infrastructure Engineering **22** (2007), 31–43.

[TTL05] D. Thain, T. Tannenbaum, and M. Livny, *Distributed computing in practice: the condor experience*, Concurrency - Practice and Experience **17** (2005), no. 2-4, 323–356.

[TYS+07] K. Tang, X. Yao, P. N. Suganthan, C. MacNish, Y. P. Chen, C. M. Chen, and Z. Yang, *Benchmark functions for the CEC'08 special session and competition on large scale global optimization*, Tech. report, Nature Inspired Computation and Applications Laboratory, USTC, China, November November 2007.

[VCC+04] L. Vanneschi, M. Clergue, P. Collard, M. Tomassini, and S. Vrel, *Fitness clouds and problem hardness in genetic programming.*, GECCO (2), Lecture Notes in Computer Science, vol. 3103, Springer, 2004, pp. 690–701.

[VdBE04] F. Van den Bergh and A. P. Engelbrecht, *A cooperative approach to particle swarm optimization*, IEEE Transactions on Evolutionary Computation **8** (2004), no. 3, 225–239.

[VH11] B. Verma and S. Z. Hassan, *Hybrid ensemble approach for classification*, Applied Intelligence **34** (2011), no. 2, 258–278.

[VHH06] T. Vanhatupa, M. Hännikäinen, and T. Hämäläinen, *Optimization of mesh WLAN channel assignment with a configurable genetic algorithm*, 1st International Workshop on Wireless Mesh (WiMeshNets'06) (Waterloo, Ontario, Canada), 2006.

[VLPD12] L. Vinh, S. Lee, Y. Park, and B. J. D'Auriol, *A novel feature selection method based on normalized mutual information*, Applied Intelligence **37** (2012), no. 1, 100–120.

[VV82] D. Van Vliet, *SATURN–A modern assignment model*, Traffic Engineering and Control **23** (1982), 578–581.

[Wil87] R. Wilcox, *New statistical procedures for the social sciences*, Hillsdale, 1987.

[Woo93] K. Wood, *Urban traffic control, systems review.*, Tech. Report PR41, TRL, Crowthorne, 1993.

[WZ07] S. Wang and J. Zhu, *Improved centroids estimation for the nearest shrunken centroid classifier*, Bioinformatics **32** (2007), no. 2, 972–979.

[XJZ+12] B. Xin, C. Jie, J. Zhang, H. Fang, and Z. Peng, *Hybridizing differential evolution and particle swarm optimization to design powerful optimizers: A review and taxonomy*, IEEE Transactions on Systems, Man, and Cybernetics, Part C **42** (2012), no. 5, 744–767.

[YKF+00] H. Yoshida, K. Kawata, Y. Fukuyama, S. Takayama, and Y. Nakanishi, *A particle swarm optimization for reactive power and voltage control considering voltage security assessment*, IEEE Transactions on Power Systems **15** (2000), no. 4, 1232–1239.

[YLL99] Xin Yao, Yong Liu, and Guangming Lin, *Evolutionary programming made faster*, Evolutionary Computation, IEEE Transactions on **3** (1999), no. 2, 82 –102.

[ZDT00] E. Zitzler, K. Deb, and L. Thiele, *Comparison of Multiobjective Evolutionary Algorithms: Empirical Results*, Evolutionary Computation **8** (2000), no. 2, 173–195.

[ZDZ12] D. Zhao, Y. Dai, and Z. Zhang, *Computational Intelligence in Urban Traffic Signal Control: A Survey*, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **42** (2012), no. 4, 485–494.

[ZJP06] H. Zhu, L. Jiao, and J. Pan, *Multi-population genetic algorithm for feature selection*, ICNC (2), 2006, pp. 480–487.

[ZT99] E. Zitzler and L. Thiele, *Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach*, IEEE Transactions on Evolutionary Computation **3** (1999), no. 4, 257–271.

[ZZLS11] Z. Zhan, J. Zhang, Y. Li, and Y. Shi, *Orthogonal learning particle swarm optimization*, IEEE Trans. Evolutionary Computation **15** (2011), no. 6, 832–847.