

On the scalability of multi-objective metaheuristics for the Software Scheduling Problem

Francisco Luna*, David L. González-Álvarez†, Francisco Chicano*, Miguel A. Vega-Rodríguez†

* *University of Málaga*

Málaga, Spain

{fb,chicano}@lcc.uma.es

† *University of Extremadura*

Cáceres, Spain

{dlga,mavega}@unex.es

Abstract—The Software Project Scheduling (SPS) problem relates to the decision of who does what during a software project lifetime. This problem has a capital importance for software companies, where the total budget and human resources involved in software development must be managed optimally in order to end up with a successful project. Companies are mainly concerned with reducing both the duration and the cost of the projects, and these two goals are in conflict with each other. A multi-objective approach is therefore the natural way of facing the SPS problem and multi-objective metaheuristics have been used to solve the problem in the past. Nowadays, software projects faced by the large companies are increasing in size and we need algorithms that are able to deal with the new large instances of the SPS problem. In this paper we analyze the scalability of four multi-objective algorithms when they are applied to the SPS problem using instances of increasing size. The algorithms are a genetic algorithm (NSGA-II), an evolution strategy (PAES), a differential evolution (DEPT) and a firefly algorithm (MO-FA). The results suggest that PAES is the algorithm with the best scalability behaviour.

Keywords—Software project scheduling, scalability analysis, multi-objective optimization

I. INTRODUCTION

As long as software projects become larger, the need to control people and processes, and to efficiently allocate resources is increasingly important. Managing such a project usually involves scheduling, planning, and monitoring tasks. Tasks may be anything from maintaining documents to typing source code, and resources include people, machines, time, etc. In this paper we focus on the assignment of employees to tasks in a software project so as to minimize two objectives: the project cost and its duration. This problem is known as the Software Project Scheduling (SPS) problem [1].

Contrary to single-objective optimization, the solution of a multi-objective problem such as SPS is not one single solution, but a set of nondominated solutions known as the *Pareto optimal set*, which is called *Pareto border* or *Pareto front* when it is plotted in the objective space [2]. Whatever solution of this set is optimal in the sense that no improvement can be reached on an objective without worsening at least another one at the same time. That is, in the context of the SPS problem, it is not possible to reduce the project cost without increasing its duration (or vice versa). The

main goal in the resolution of a multi-objective problem is to compute the set of solutions within the Pareto optimal set and, consequently, the Pareto front. In order to deal with multi-objective optimization problems, metaheuristic algorithms [3] have been widely used. Indeed, the most well-known algorithms in the multi-objective community fall into this kind of search technique [4].

Previous works in the literature have addressed the multi-objective SPS problem with metaheuristics [5], where 36 instances with up to 30 tasks and 15 employees have been tackled. However, as long as software projects become larger and larger, the number of tasks and resources involved might be much greater as well. This is a major issue that has to be taken into consideration and poses new challenges to the optimization algorithms because the search space to be explored is also much larger [6]. Our contribution in this work is to evaluate the scalability capabilities of four multi-objective metaheuristics, two classical methods—NSGA-II [7] and PAES [8]—plus two recent algorithms—DEPT [9] and MO-FA [10]—on a set of 36 instances with an exponential increase in both the number of tasks (from 16 to 512) and the number of employees (from 8 to 256). We selected NSGA-II and PAES because they were the best algorithms in previous work [5] and we selected DEPT and MO-FA because they have been never applied to this problem. Two quality indicators, the hypervolume (HV) [11] and the attainment surfaces [12], have been used to measure the quality of the resulting Pareto fronts.

The paper is structured as follows. The next section provides the reader with the formulation of the SPS problem. Section III briefly describes the four multi-objective metaheuristics used. The experimentation performed to assess the performance of these algorithms is detailed in Section IV. Section V includes the main conclusions of this work and devises the future lines for further research.

II. THE SPS PROBLEM

We follow here the same formulation proposed in [1]. Thus, the resources considered are people with a set of skills and a salary. These employees have a maximum degree of dedication to the project. Formally, each person (employee) is denoted with e_i , where i goes from 1 to E (the number

of employees). Let SK be the set of skills, and s_i the i -th skill with i varying from 1 to $S = |SK|$. The skills of the employee e_i will be denoted with $e_i^{skills} \subseteq SK$, the monthly salary with e_i^{salary} , and the maximum dedication to the project with e_i^{maxded} . The salary and the maximum dedication are real numbers. The former is expressed in abstract currency units, while the latter is the ratio between the amount of hours dedicated to the project and the full working day length of the employee. The tasks are denoted with t_i , where i goes from 1 to T (the number of tasks). Each task t_i has a set of required skills associated with it, which we denote with t_i^{skills} , plus an effort t_i^{effort} , expressed in person-month (PM). The tasks must be performed according to a Task Precedence Graph (TPG) that indicates which tasks must be completed before a new task is started. The TPG is an acyclic directed graph $G(V, A)$ with a vertex set $V = \{t_1, t_2, \dots, t_T\}$ and an arc set A , where $(t_i, t_j) \in A$ if the task t_i must be completed, with no other intervening tasks, before task t_j can start. The objectives of the SPS problem are to minimize the cost and the duration of the project. The constraints are (1) that each task must be performed by at least one person, (2) the set of required skills of a task must be included in the union of the skills of the employees performing the task, and (3) no employee must exceed her/his maximum dedication to the project.

A solution can be represented with a matrix $\mathbf{X} = (x_{ij})$ of size $E \times T$, where $x_{ij} \geq 0$. The element x_{ij} is the degree of dedication of the employee e_i to the task t_j . In order to compute the project duration, denoted with p_{dur} , we need to calculate the duration of each individual task (t_j^{dur}). This is calculated in the following way:

$$t_j^{dur} = \frac{t_j^{effort}}{\sum_{i=1}^E x_{ij}} \quad (1)$$

The next step is to compute the starting and ending time for each task (t_j^{start} and t_j^{end}), which are defined according to the following expressions:

$$t_j^{start} = \begin{cases} 0 & \text{if } \nexists t_i, (t_i, t_j) \in A \\ \max_{t_i, (t_i, t_j) \in A} \{t_i^{end}\} & \text{otherwise} \end{cases} \quad (2)$$

$$t_j^{end} = t_j^{start} + t_j^{dur} \quad (3)$$

The project duration, p_{dur} , is the maximum ending time ever found: $p_{dur} = \max_{j=1}^T \{t_j^{end}\}$.

The project cost p_{cost} is the sum of the salaries paid to the employees for their dedication to the project. These charges are computed by multiplying the salary of the employee by the time spent on the project. The time spent on the project is the sum of the dedication multiplied by the duration of each task. In summary:

$$p_{cost} = \sum_{i=1}^E \sum_{j=1}^T e_i^{salary} \cdot x_{ij} \cdot t_j^{dur} \quad (4)$$

A solution is feasible if it satisfies the following constraints (otherwise it is unfeasible):

$$\sum_{i=1}^E x_{ij} > 0 \quad \forall j \in \{1, 2, \dots, T\} \quad (5)$$

$$t_j^{skills} \subseteq \bigcup_{\{i|x_{ij}>0\}} e_i^{skills} \quad \forall j \in \{1, 2, \dots, T\} \quad (6)$$

$$\sum_{i=1}^E e_i^{over} = 0 \quad (7)$$

where e_i^{over} is the overwork of employee e_i , computed as:

$$e_i^{over} = \int_{\tau=0}^{\tau=p_{dur}} ramp(e_i^{work}(\tau) - e_i^{maxded}) d\tau \quad (8)$$

$$e_i^{work}(\tau) = \sum_{\{j|t_j^{start} \leq \tau \leq t_j^{end}\}} x_{ij} \quad (9)$$

In the previous expression $ramp(x)$ is a function which is x if $x > 0$ and 0 otherwise.

III. ALGORITHMS

A. NSGA-II

The NSGA-II algorithm was proposed by Deb *et al.* [7]. It is a genetic algorithm based on obtaining a new population from the original one by applying the typical genetic operators (selection, crossover, and mutation); then, the individuals in the new and old population are sorted according to their rank, and the best solutions are chosen to create a new population. In case of having to select some individuals with the same rank, a density estimation based on measuring the crowding distance to the surrounding individuals belonging to the same rank is used to get the most promising solutions.

B. PAES

PAES is a metaheuristic proposed by Knowles and Corne [8]. The algorithm is based on a simple (1+1) evolution strategy. To find diverse solutions in the Pareto optimal set, PAES uses an external archive of nondominated solutions, which is also used to decide about the new candidate solutions. An adaptive grid is used as a density estimator in the archive. We have used a real coded version of PAES, applying a polynomial mutation operator.

C. Differential Evolution with Pareto Tournaments

The Differential Evolution (DE) is an evolutionary algorithm created by Ken Price and Rainer Storn [9]. The fundamental idea behind DE is a scheme for generating new possible solutions (trial individuals) taking advantage of the differences among the population (target individuals), according to its simple formulae of vector-crossover and mutation. We have defined a new multiobjective version that incorporates the Pareto Tournaments concept (DEPT), choosing the best solution between two given ones (in this

case, the target and the trial individuals). To do this we calculate a multiobjective fitness value (MOF) for each individual by using the following equation:

$$MOF(i) = IsDominated(i) * PS + Dominates(i) \quad (10)$$

where i is the processed individual and PS the population size. In Equation (10) we consider the number of solutions that dominate the individual and the number of solutions of the population that are dominated by it. If both individuals (trial and target) obtain the same MOF, we have two solutions of the same Pareto front and we have to apply another selection criterion: the crowding distance. In this case, the individual with greater value of the crowding distance will be the winner of the Pareto Tournament.

D. Multiobjective Firefly Algorithm

The Firefly Algorithm (FA) is one of the latest nature-inspired optimizers proposed. This algorithm is defined by Xin-She Yang [10] and it is inspired by the flash pattern and characteristics of fireflies. To solve the SPS problem we have developed the Multiobjective Firefly Algorithm (MO-FA). In the simplest form, the light intensity $I(r)$ varies according to a fixed absorption coefficient in the media, γ . So, the brightness decreases as $e^{-\gamma r_{ij}^2}$. Therefore, in this algorithm the two most important factors are the variation of the light intensity and the formulation of the attractiveness. For simplicity, the attractiveness of a firefly is determined by its brightness, which is associated with the objective functions. Consequently, each firefly is attracted to another more attractive (brighter) by the following equation:

$$x_i = x_i + \beta_0 e^{-\gamma r_{ij}^2} (x_j - x_i) + \alpha \left(rand() - \frac{1}{2} \right) \quad (11)$$

while the second term of Equation (11) is due to the attraction, the third term brings randomness to the process with the control parameter α . The parameter values have been established as proposed in [10].

IV. EXPERIMENTATION

This section is aimed at presenting the experiments conducted to evaluate the scalability capabilities of the previously described algorithms on 36 instances of the SPS problem.

A. Methodology

In order to measure the performance of the multi-objective solvers used here, the quality of their resulting nondominated set of solutions has to be considered. Two indicators have been used for this purpose in this work: the hypervolume (HV) [11] and the attainment surfaces [12].

The HV is considered as one of the more suitable indicators in the multi-objective community since it provides

a measure that takes into account both the convergence and diversity of the obtained approximation set. Higher values of the hypervolume metric are desirable. Since this indicator is not free from an arbitrary scaling of the objectives, we have built up a reference Pareto front (RPF) for each problem composed of all the nondominated solutions found for each problem instance by all the algorithms. Then, the RPF is used to normalize each approximation prior to compute the HV value by mapping all the nondominated solutions to $[0, 1]$. This way the reference point to compute the HV values is (1,1), which results from the mapping of the extreme solutions of the RPF.

While the HV allows one to numerically compare different algorithms, from the point of view of a decision maker, knowing about the HV value might not be enough, because it gives no information about the shape of the front. The empirical attainment function (EAF) [12] has been defined to do so. EAF graphically displays the expected performance and its variability over multiple runs of a multi-objective algorithm. In short, the EAF is a function α from the objective space \mathbb{R}^n to the interval $[0, 1]$ that estimates for each vector in the objective space the probability of being dominated by the approximated Pareto front of one single run of the multi-objective algorithm. Given the r approximated Pareto fronts obtained in the different runs, the EAF is defined as:

$$\alpha(z) = \frac{1}{r} \sum_{i=1}^r I(A^i \preceq \{z\}) \quad (12)$$

where A^i is the i -th approximated Pareto front obtained with the multi-objective algorithm and I is an indicator function that takes value 1 when the predicate inside it is true, and 0 otherwise. The predicate $A^i \preceq \{z\}$ means A^i dominates solution z . Thanks to the attainment function, it is possible to define the concept of $k\%$ -attainment surface [12]. The attainment function α is a scalar field in \mathbb{R}^n and the $k\%$ -attainment surface is the level curve with value $k/100$ for α . Informally, the 50%-attainment surface in the multi-objective domain is analogous to the median in the single-objective one.

Metaheuristics are stochastic algorithms; therefore the results have to be provided with statistical significance. The following statistical procedure has been used. First, 30 independent runs for each algorithm and each problem instance have been performed. The HV indicator and the attainment surfaces are then computed. In the case of HV, a multiple comparison test has been carried out in order to check if the differences are statistically significant or not. All the statistical tests are performed with a confidence level of 95%.

B. Parameterization

In order for a fair comparison among all the algorithms to be performed, they all are required to run for 100,000 func-

tion evaluations and to obtain 100 nondominated solutions at most. NSGA-II uses a population size of 100 individuals, whereas both DEPT and MO-FA manipulate 32 solutions (PAES has one single solution since it is a (1+1)-evolution strategy).

A solution to the problem is a vector of floating point numbers in which the component i stores the dedication of employee $\lfloor i/T \rfloor$ to task $i \bmod T$ (where T is the number of tasks). With this encoding, the typical operators from the multi-objective metaheuristic community have been used. So therefore, the NSGA-II has adopted simulated binary crossover (SBX) and polynomial mutation. The distribution indices 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. In PAES we have also used a polynomial mutation operator, with the same distribution index as indicated before. DEPT has been configured with the RandToBest/1/Binomial scheme, $CR = 0.9$ and $F = 0.5$. The mutation factor for MO-FA is 0.5.

C. Repair Operator

Because of the size of the SPS instances addressed (see next section), it has been very hard for all the algorithms to compute feasible solutions. We found out that the employees' overwork (Equation (7)) is the most difficult constraint to meet. The reason is that the search operators are not endowed with problem-specific knowledge so it is usual to find assignments in which one or more employees exceed their maximum dedication. In order to deal with such issue, we have used a repair operator that, whenever an overwork in the assignment is detected, it is fixed by dividing the dedication of all the employees to all the tasks by the maximum overwork of the employees. That is, the effect of the operator is:

$$x'_{ij} = \frac{x_{ij}}{\max_{i,\tau} \{e_i^{work}(\tau)\} + \varepsilon} \quad (13)$$

where $\varepsilon = 0.00001$ is used in order to prevent from inaccuracies in the floating-point operations.

This operator increases the project duration of the tentative solution and keeps the cost unchanged. That is: $p'_{dur} = p_{dur} \cdot (\max_{i,\tau} \{e_i^{work}(\tau)\} + \varepsilon)$ and $p'_{cost} = p_{cost}$. In addition, the new solution satisfies the third constraint (the one related to the overwork). From the point of view of the algorithmic complexity, the overhead introduced is the same as the evaluation of the solution, since the coefficient used in the denominator is computed at the same time that the solution is evaluated. If a solution violates the other constraints of the problem, then it is penalized in the selection and replacement operators (it is the last one selected).

Table I
RANKING OF THE MO SOLVERS BASED ON THEIR HV VALUES.

	NSGA-II	PAES	DEPT	MO-FA
i16-8	0.661±0.028	0.732±0.019	0.311±0.020	0.540±0.022
i16-16	0.468±0.026	0.826±0.013	0.327±0.038	0.608±0.031
i16-32	0.147±0.016	0.809±0.009	0.226±0.037	0.379±0.088
i16-64	0.129±0.017	0.858±0.010	0.287±0.034	0.370±0.092
i16-128	0.048±0.013	0.722±0.012	0.122±0.028	0.139±0.075
i16-256	0.018±0.008	0.682±0.010	0.078±0.029	0.069±0.029
i32-8	0.538±0.029	0.721±0.017	0.107±0.024	0.209±0.029
i32-16	0.190±0.026	0.820±0.012	0.035±0.017	0.341±0.045
i32-32	0.121±0.014	0.743±0.009	0.109±0.018	0.284±0.042
i32-64	0.049±0.010	0.795±0.025	0.042±0.018	0.263±0.083
i32-128	0.041±0.007	0.726±0.012	0.080±0.015	0.061±0.011
i32-256	0.009±0.007	0.617±0.016	0.011±0.011	0.000±0.000
i64-8	0.463±0.031	0.813±0.014	0.089±0.014	0.140±0.020
i64-16	0.221±0.022	0.959±0.011	0.026±0.007	0.347±0.031
i64-32	0.063±0.011	0.798±0.008	0.001±0.002	0.321±0.031
i64-64	0.073±0.012	0.870±0.006	0.004±0.005	0.032±0.023
i64-128	0.027±0.007	0.738±0.008	0.006±0.005	0.002±0.004
i64-256	0.000±0.000	0.618±0.013	0.000±0.000	0.000±0.000
i128-8	0.320±0.019	0.986±0.006	0.003±0.006	0.076±0.031
i128-16	0.253±0.023	0.988±0.013	0.000±0.000	0.000±0.001
i128-32	0.213±0.016	0.980±0.010	0.000±0.000	0.000±0.000
i128-64	0.095±0.008	0.920±0.021	0.000±0.000	0.044±0.023
i128-128	0.029±0.008	0.782±0.009	0.000±0.000	0.000±0.000
i128-256	0.008±0.005	0.514±0.017	0.000±0.000	0.000±0.000
i256-8	0.309±0.015	0.998±0.002	0.000±0.000	0.010±0.019
i256-16	0.156±0.012	0.987±0.007	0.000±0.001	0.281±0.018
i256-32	0.089±0.009	0.927±0.012	0.000±0.000	0.009±0.012
i256-64	0.028±0.005	0.739±0.022	0.000±0.000	0.002±0.004
i256-128	0.019±0.005	0.752±0.010	0.000±0.000	0.000±0.000
i256-256	0.003±0.003	0.626±0.016	0.000±0.000	0.000±0.000
i512-8	0.235±0.016	0.995±0.002	0.000±0.000	0.029±0.025
i512-16	0.103±0.007	0.960±0.011	0.000±0.000	0.025±0.024
i512-32	0.029±0.008	0.961±0.010	0.000±0.000	0.000±0.000
i512-64	0.018±0.007	0.820±0.014	0.000±0.000	0.000±0.000
i512-128	0.198±0.004	0.820±0.010	0.102±0.003	0.105±0.006
i512-256	0.008±0.005	0.560±0.031	0.000±0.000	0.000±0.000

D. SPS Instances

For the empirical study we have used a total of 36 instances¹. Each instance represents a different software project. The number of employees and tasks scales up from 16 to 512 and from 8 to 256, respectively. The total number of skills in the project, S , is 10 and the number of skills per employee ranges from 6 to 7. We denote the instances with $iT-E$, where T and E are the number of tasks and employees, respectively. For example, the instance i128-32 has 128 tasks and 32 employees. The maximum dedication for all the employees is 1 (full working day) in the 36 instances.

E. Results

This first part of the analysis is devoted to compare the multi-objective metaheuristics on the set of 36 SPS instances by using the HV indicator. Table I shows the median and intercuartile range of the HV values of the algorithms for each instance on 30 independent runs. The gray coloured background in a table cell has been used to better show the best performing algorithm.

The HV values draw a clear scenario: PAES has been able to approximate the Pareto fronts with the best (highest

¹mstar.lcc.uma.es/problems/swscheduling.html

indicator values and with significant differences. This means that, considering both convergence and diversity, PAES has clearly outperformed NSGA-II, DEPT, and MO-FA. For the instances up to 64 tasks, MO-FA has been the second best algorithm but, for larger instances, NSGA-II has reached higher (better) HV values than MO-FA and DEPT. This latter algorithm has performed the worst in the comparison undertaken in this work. We want to clarify two points here. First, the zero HV values obtained by DEPT and MO-FA are due to the normalization process that simply discards the non-dominated solutions that are out of the limits of the RPF built up for each instance. And, second, the statistical analysis carried out to provide the results with confidence has reported that it always exists statistical difference between PAES and the other three algorithms (not shown because of room constraints).

In order to better explain these results, Figure 1 displays the 50%-attainment surfaces of four SPS instances: i32-8, i32-16, i32-32, and i32-64. These plots easily justify the high HV value obtained by PAES. Indeed, its approximated Pareto fronts are not only near to the RPF (specially for those solutions with low project costs) but also they cover a larger region in the objective space. This fact becomes more evident as long as the instances are larger. We can therefore conclude that PAES has the desirable property of scalability, which is precisely the aim of this study. We can provide an explanation for such a behavior. It has to do with the effect of the repair operator used and the disruption provoked by the recombination operator used in NSGA-II and DEPT. Indeed, once PAES has reached the feasible region for a given instance, its search engine is based on a mutation operator that is changing, on average, one single assignment of an employee to a task ($p_m = 1/L$, where L is the length of the solution). As a consequence, it is rather easy to keep the exploration within the feasible region and there is no need for the repair operator to be applied. On the other hand, when trying to recombine solutions (either with SBX in NSGA-II or the differential evolution crossover in DEPT), the newly generated assignments often incur in employees' overwork. The repair operator then fixes such unfeasibility by decreasing the employees' dedication, which leads to an usually long increment in the project duration but keeping its cost almost the same. This is why it is hard for NSGA-II and DEPT to explore the same region of the search space as PAES.

As to the scalability capabilities of the algorithms, let's start by analyzing their behavior with an increasing number of employees (i.e., within each section in which Table I has been split). NSGA-II has obtained its best (highest) HV values for instances with 8 employees, that is, iX-8, where $X = \{16, 32, \dots, 512\}$. Then, the higher the number of employees, the lower (worse) the HV value. This is due to the effect of the crossover operator on larger solutions. The sequence of plots in Figure 1 shows this fact. It can

be seen that as the instance size gets larger (from 8 to 64 employees), the portion of the 50%-attainment surface of NSGA-II becomes narrower and so the HV value gets lower as well. A similar behavior is reported for DEPT and MO-FA, which have computed the approximated Pareto fronts that are further from the RPF. In the case of PAES, it can be observed that its HV values start decreasing, in general, in instances with over 128 employees.

With respect to the scalability in terms of the increasing number of tasks, it is clear that it makes the SPS instances harder, specially for DEPT and MO-FA (for example, many zero HV median values are obtained for DEPT). PAES, again, is clearly the algorithm that better scales with the instance size.

Finally, regarding wall clock time, the execution of the algorithms requires between a few seconds in the case of the smallest instances and around 5 hours in the case of the largest instances.

V. CONCLUSIONS AND FUTURE WORK

In this paper we have analyzed the scalability of four different multi-objective metaheuristic algorithms when they are applied to the Software Project Scheduling problem. The efficient resolution of the problem is important in the context of large software companies, which have to deal with large software projects. In the experimental evaluation we used a benchmark composed of 36 automatically generated instances with increasing size. The results clearly suggest that PAES is the best algorithm in terms of quality of the solutions and scalability. The second best algorithm seems to be MO-FA for small instances and NSGA-II for large instances.

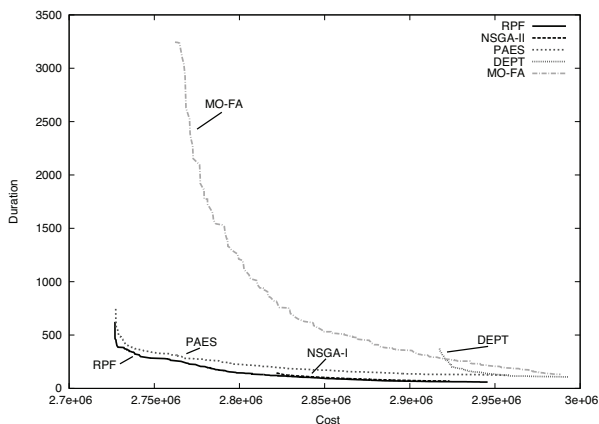
A future line of research would be the design of a new version of the problem including some real-world issues that are not present in the current formulation, like communication overhead in a group of employees or solution robustness. It is also possible to add new multi-objective algorithms to the scalability comparison. Especially interesting is the design of new hybrid algorithms that combine together the best features of the algorithms analyzed in this work.

ACKNOWLEDGMENT

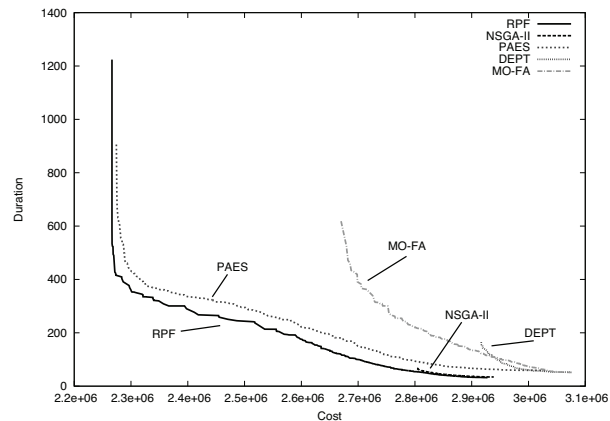
This work has been partially funded by the Spanish Ministry of Science and Innovation and FEDER under contract TIN2008-06491-C04. It has also been partially funded by the Andalusian Government under contract P07-TIC-03044. Thanks also to the Fundación Valhondo, for the economic support offered to David L. González-Álvarez to make this research.

REFERENCES

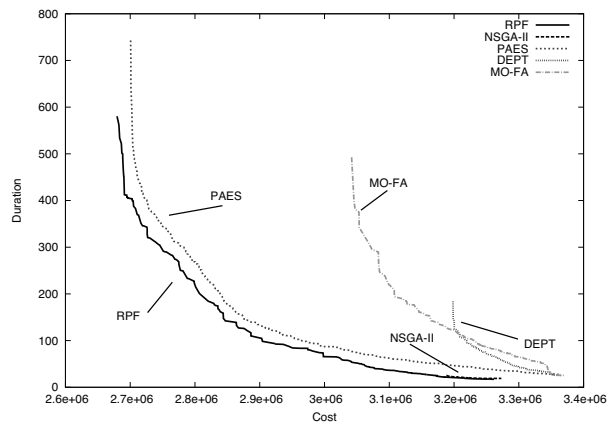
- [1] E. Alba and F. Chicano, "Software project management with GAs," *Information Sciences*, vol. 177, no. 11, pp. 2380–2401, June 2007.



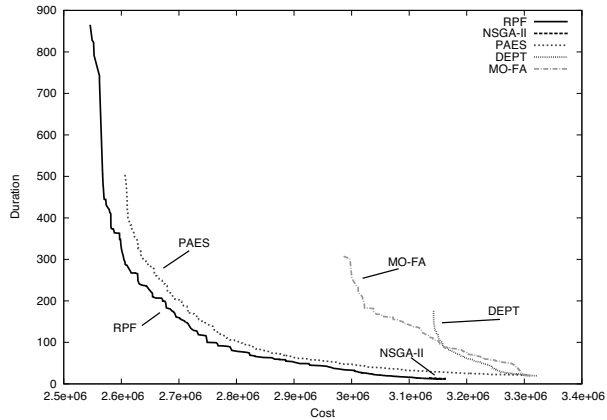
(a) i32-8



(b) i32-16



(c) i32-32



(d) i32-64

Figure 1. 50%-Attainment surfaces of the algorithms in four instances with 32 tasks.

- [2] K. Deb, *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, 2001.
- [3] F. W. Glover and G. A. Kochenberger, *Handbook of Metaheuristics*. Kluwer, 2003.
- [4] C. A. Coello Coello, G. B. Lamont, and D. A. Van Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*, 2nd ed. New York: Springer, 2007.
- [5] J. F. Chicano, F. Luna, A. J. Nebro, and E. Alba, "Using multi-objective metaheuristics to solve the software project scheduling problem," in *Proceedings of GECCO*, 2011, pp. 1915–1922.
- [6] J. J. Durillo, A. J. Nebro, C. A. Coello, J. García-Nieto, F. Luna, and E. Alba, "A study of multiobjective metaheuristics when solving parameter scalable problems," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 4, pp. 618 – 635, 2010.
- [7] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. on Ev. Comp.*, vol. 6, no. 2, pp. 182–197, 2002.
- [8] J. Knowles and D. Corne, "Approximating the nondominated front using the pareto archived evolution strategy," *Evolutionary Computation*, vol. 8, no. 2, pp. 149 – 172, 2000.
- [9] K. Price and R. Storn, "Differential evolution - a simple evolution strategy for fast optimization," *Dr. Dobb's Journal*, vol. 22, no. 4, pp. 18–24 and 78, 1997.
- [10] X.-S. Yang, "Firefly algorithms for multimodal optimization," in *5th International Symposium of Stochastic Algorithms: Foundations and Applications (SAGA'09)*, vol. 5792, 2009, pp. 169 – 178.
- [11] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.
- [12] J. Knowles, "A summary-attainment-surface plotting method for visualizing the performance of stochastic multiobjective optimizers," in *5th Int. Conf. on Intelligent Systems Design and Applications (ISDA'05)*, 2005, pp. 552 – 557.