In ACDM'98, 21–23 April 1998, Plymouth, UK, pp. 255–267

Multi-objective Optimisation and Preliminary Airframe Design

Dragan Cvetković* and Ian Parmee†
Plymouth Engineering Design Centre
University of Plymouth, UK

Eric Webb
British Aerospace
Warton, UK

Abstract

In this paper we explore established methods for optimising multi-objective functions whilst addressing the problem of preliminary design. Methods from the literature are investigated and new ones introduced. All methods are evaluated within a collaborative project with British Aerospace for whole system airframe design and the basic problems and difficulties of preliminary design methodology are discussed.

Our Genetic Algorithm is expanded to integrate different methods for optimising multi–objective functions. First, methods based on scalarisation and the utilisation of weights are addressed. Methods based on Pareto order are then analysed and two different sorting techniques (dominated/non–dominated and Pareto rank) investigated. Finally, several variants of sub–population based algorithms are presented.

All presented methods are also analysed in the context of whole system design, discussing their advantages and disadvantages.

1 Introduction

When dealing with industrial design problems, one immediately realises that there are significant differences between so called 'textbook optimisations problems' and 'real world applications'. In both cases, when attempting multi–objective optimisation, we have a function to optimise:

Definition 1 Let $n > 0$, $k > 0$, $\mathcal{D} = \mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_n \subseteq \mathbf{R}^n$, and $\mathcal{R} = \mathcal{Y}_1 \times \mathcal{Y}_2 \times \ldots \times \mathcal{Y}_k \subseteq \mathbf{R}^k$. Let further $f_i : \mathcal{D} \mapsto \mathcal{Y}_i$ for $1 \leq i \leq k$ and finally $\mathbf{F} : \mathcal{D} \mapsto \mathcal{R}$, so that $\mathbf{F}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x}))$.

Our goal is to optimise function $\mathbf{F}(\boldsymbol{x})$ under additional constraints i.e.

$$\max_{\boldsymbol{x}} \mathbf{F}(\boldsymbol{x}) \tag{1}$$

$$g_1(\boldsymbol{x},\boldsymbol{p}) \leq 0, \ldots, g_l(\boldsymbol{x},\boldsymbol{p}) \leq 0 \tag{2}$$

*e–mail: D.Cvetkovic@plymouth.ac.uk
†e–mail: I.Parmee@plymouth.ac.uk

where $\boldsymbol{p} = (p_1, \ldots, p_u)$ are additional (real–valued) parameters. The problem itself is not easy because we have many optimisation criteria, some of which contradict one another. This problem is well known and a number of non–genetic [1, 2] and genetic algorithm approaches exists [3, 4, 5]. An additional problem is that not all objectives are equally important which necessitates the use of weights [6] and preferences [7].

However, initial discussion with the industrial partner makes the problem even harder due to the following additional (meta–)objectives:

- We have objectives and we have constraints. The difference between them is very fuzzy and some of them will move from objectives to constraints or vice versa. Some constraints are hard, some not; some will change or disappear whilst others may be introduced as the problem knowledge base expands.
- In many cases the variable ranges are also fuzzy and there is a requirement for exploration outside of the default regions.
- The output should contain both optimal solutions and suggestions of extending ranges and/or inclusion/removals of constraints.
- A set of results is required which the engineer can analyse off-line.
- The end–user (designer) should not be confused with the number of parameters and possibilities the (optimisation) program offers, we must avoid cognitive overload.
- The engineer wishes to interact with the search process by sampling results after N functions evaluations, and adapting parameters and/or constraints.

The problems of conceptual design relate to the fuzzy nature of initial design concepts and the many different variants that engineer wishes to try. Computers should be able to help him exploration of those variants whilst also suggesting some others as well. This problem has been investigated in Plymouth Engineering Design Centre (PEDC) before [8, 9]. The schema of computer aided design is simply presented in Figure 1.1: at the beginning, the computer system and the designer might have totally different concepts about the object. Ideally they should agree (after some 'negotiations') and have the same idea at the end of the process.

1.1 The BAe model

Our project is based on a collaboration with British Aerospace (BAe) and utilises their CAPS system. CAPS (Computer aided project studies) is an integrated computer software suite, developed at BAe for use by engineers and designers during the earliest investigation stages of a new aircraft project.

The scope of disciplines covered by CAPS is wide: preliminary geometric definition, aerodynamic analysis, mass estimation, performance analysis, cost estimation etc.

In a typical job the user programs CAPS to search for design solutions that meet performance requirements whilst satisfying a number of constraints.

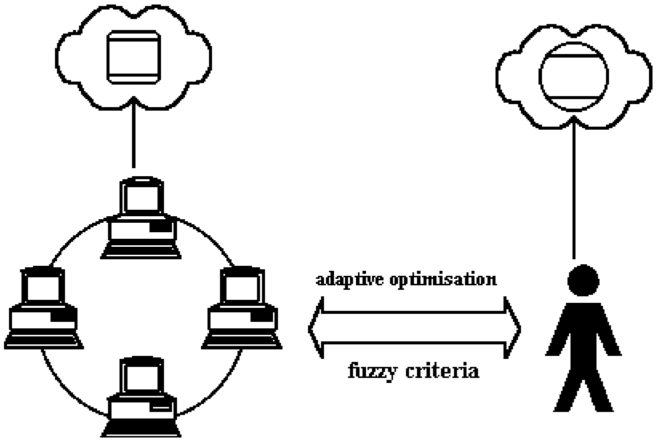

Figure 1.1: The initial phase of computer aided whole system design

BAe has developed a MINICAPS model [10] based upon the full CAPS representation for use by the PEDC. At the moment there are 8 input and 9 output parameters (i.e. $n = 8$ and $k = 9$ in Definition 1). All constraints so far are variable domain. Input and output parameters of the function are presented in Figure 1.2. In the further text we will usually denote input variables by $x_1, \ldots, x_8$ and the objectives $y_1, \ldots, y_9$.

The project commenced in April 1997, and this paper mainly describes the initial phase and a comparison of the integration of well–known multi–objective techniques with MINICAPS.

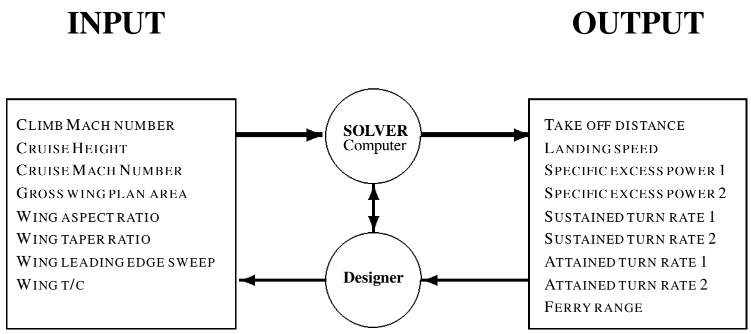

Figure 1.2: Input and output parameters of the BAe function and designer interaction.

2 Optimisation methods

2.1 Weighted sums

Weighted sums is a method of scalarisation of vector functions [1, p.32], [2].

Definition 2 For a function $\mathbf{F}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x}))$ from Definition 1 and a vector $\boldsymbol{w} = (w_1, \ldots, w_k)$, so that $\sum_{i=1}^{k} w_i = 1$, define

$$\mathbf{F}_{\boldsymbol{w}}(\boldsymbol{x}) = \sum_{i=1}^{k} w_i \cdot f_i(\boldsymbol{x}) \tag{3}$$

Usually one assigns weights according to the importance of objectives: more important objective will get a higher weight, less important objectives will get lower weights. However, since not all the objectives have the same range of values, they must further be normalised.

One method for choosing weights is by calculating $w_i^* = 1/\max_{\boldsymbol{x}} f_i(\boldsymbol{x})$. The vector $(\max_{\boldsymbol{x}} f_i(\boldsymbol{x}))_{i=1,k}$ is called the *positive ideal solution* [6, p.90]. If we are to minimise an objective y, there are two approaches:

1. Use identity $\min y = -\max(-y)$. However, the genetic algorithm has a problem with negative fitness values — that problem can be avoided by maximising $M - \max(-y)$ for a large enough M. More precisely, if objective y is positive and bounded i.e. $y \in [m, M]$, then $0 \leq (M-y)/(M-m) \leq 1$ and value $1/(M-m)$ can be used for the weight.

2. Assuming that $y > 0$, use identity $\min y = 1/\max(1/y)$ and so maximise $1/y$. In this case we have no problems estimating weights, but we can't always know in advance if the objective y is non–zero in all cases.

Example 1 The BAe function described above is $\mathbf{F} : \mathbf{R}^8 \mapsto \mathbf{R}^9$. Objectives 1 and 2 should be minimised and the rest of the objectives maximised. If we optimise for one objective only (i.e. just ignore the other 8 output values), we get the optima and weights w_i^* presented in Table 2.1.

i	1	2	3	4	5	6
$\mathrm{opt}_{\boldsymbol{x}} f_i$	217.800	56.8903	148.176	82.9057	16.6629	10.5922
w_i^*	-0.00108	-0.01186	0.00675	0.01206	0.06001	0.09441

i	7	8	9
$\mathrm{opt}_{\boldsymbol{x}} f_i$	27.8898	18.6173	10263.7
w_i^*	0.03586	0.05371	0.00010

Table 2.1: Results obtained by optimising BAe function one objective at the time

Using results from Table 2.1, optimising the function

$$\mathbf{F}_{\boldsymbol{w}^*}(\boldsymbol{x}) = \sum_{i=1}^{9} w_i^* \cdot f_i(\boldsymbol{x})$$

gives us the following as an optimum solution:

$$\mathbf{F}_{w^*}(0.86, 9033.6, 0.9, 56.04, 2.42, 0.1, 58.46, 0.03) = 4.027$$
$$\mathbf{F}(0.86, 9033.6, 0.9, 56.04, 2.42, 0.1, 58.46, 0.03) =$$
$$(259.97, 66.386, 114.75, 37.443, 12.578, 8.4048, 20.771, 18.617, 5761.2)$$

There are several drawbacks with this method. It is very computationally expensive, since for n objectives, we have to perform at least n GA runs (one for each objective) *before* we can start to optimise our function. This might make sense if we are not going to change (add or delete) our objectives and constraints: in that case we can just reuse those values. However, in the conceptual design phase, the probability of objectives and constraints variation is high, and thus the fitness landscape will change therefore, necessitating the re–calculation of our weights again and again. So, this method is not very feasible for our purpose. We will however use it in the further text in order to demonstrate different techniques and results obtained using them.

2.2 Lexicographic method

Lexicographic method for multi-objective optimisation is based on lexicographic ordering [11]:

Definition 3 We will say that point $\boldsymbol{x} \in \mathbf{R}^n$ is *less in lexicographic order* then the point $\boldsymbol{y} \in \mathbf{R}^n$, written $\boldsymbol{x} \preceq \boldsymbol{y}$ if either $\boldsymbol{x} = \boldsymbol{y}$ or the first nonzero component of $\boldsymbol{x} - \boldsymbol{y}$ is negative.

Obviously, since the sorting of individuals is done using relation $\preceq$, the first objective is the most important one, the second objective is a bit less important etc. so one has to be very aware of the importance of the particular objectives. According to [11]:

> Lexicographic optimization arises in those practical situations where optimal policies are determined by making decisions *successively*. First the most important objectives are met. Among the solutions which meet this objective a smaller set is then chosen to satisfy an objective second in importance etc.

So this method is quite opposite to Pareto sorting method, presented below, that treats all objectives equally.

However, there are some obvious drawbacks using this method, especially in preliminary system design where the designer doesn't always know the importance of particular objective. What is usually done in that case is to use *random lexicographic order*: the order of objectives we sort on is chosen randomly and varies from generation to generation. We have implemented this method within a GA and some results are presented in Table 2.3 on page 9.

2.3 Pareto optimisation

Consider a vector function $\mathbf{F} : \mathcal{D} \mapsto \mathcal{R}$ from Definition 1 again.

Definition 4 We will say that a point $\boldsymbol{x} \in \mathcal{D}$ *Pareto–dominates* a point $\boldsymbol{y} \in \mathcal{D}$ with respect to function $\mathbf{F}$, denoted $\boldsymbol{y} \leq^P_{\mathbf{F}} \boldsymbol{x}$, if $\wedge_{i=1}^{k}(f_i(\boldsymbol{y}) \leq f_i(\boldsymbol{x}))$ and at least one of inequalities is strict.

We say that point $\boldsymbol{x}_P \in \mathcal{D}$ is *Pareto–optimal* or *non–dominated* (for a given function $\mathbf{F}$) if there is no point $\boldsymbol{y} \in \mathcal{D}$ that Pareto–dominates $\boldsymbol{x}$ i.e. $(\neg\exists \boldsymbol{y} \in \mathcal{D})(\boldsymbol{x}_P \leq^P_{\mathbf{F}} \boldsymbol{y})$

Set $\mathcal{F} \subseteq \mathcal{D}$ is called the *Pareto front* with respect to function $\mathbf{F}$ if every element $\boldsymbol{x} \in \mathcal{F}$ is Pareto optimal with respect to function $\mathbf{F}$. In other words, the Pareto front is the maximal set of non–dominated elements.

In the following text we will drop indices $\mathbf{F}$ and P and will just write $\leq$ instead of $\leq^P_{\mathbf{F}}$.

Example 2 In Figure 2.1 we can see an example of a partial order. All elements at the top of the Figure (i.e. points A, B, C, D and E (!)) are non–dominated and therefore in the Pareto front.

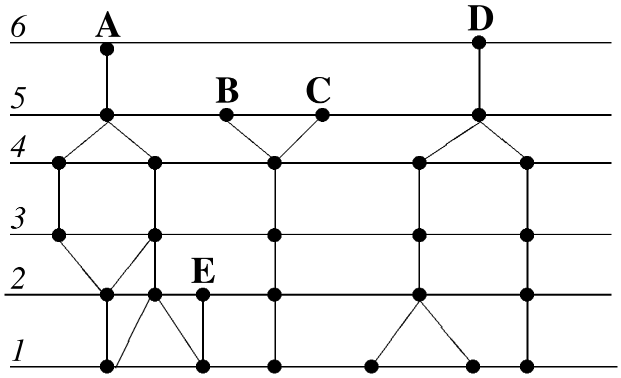

Figure 2.1: Example of partial order and ranking.

Example 2 also shows that not all elements from the Pareto front are automatically of a good fitness value (e.g. point E). Point E can be in Pareto front if one of the objective values is very high (higher that any other) but all others are low.

We have used Pareto method in Genetic Algorithms in the selections phase in the following way:

Pareto tournament Use standard tournament selection but the chosen individual will be the best according to Pareto ordering;

Pareto sort Sort first according to Pareto ordering and then, if both individuals are dominant (or non dominant), sort according to the fitness value;

Pareto truncation Choose parents only among the non–dominated parents.

It is interesting to observe how the number of non-dominated elements in the population increases. Figure 2.2, presents a diagram of number of non–dominated elements in population versus generation number. Population size used is 100. We have used standard tournament of size 2 as a selection method. The results are averaged over 50 runs and standard deviation is presented using error bars. We were optimising on y_3, y_4 and y_9 simultaneously. What we can see immediately is that after some 150 generations more then 50% of the populations is formed by non–dominated individuals and that the selection pressure gradually decreases with the number of generations.

In the case of conceptual design, this also means that the Pareto front, the set the designer is usually interested in, will be much too large for a human to handle. Even with a small population of 100 individuals, after 400 generations, the size of the Pareto front increases to 80 i.e. 80 solutions (more or less different) to deal with.

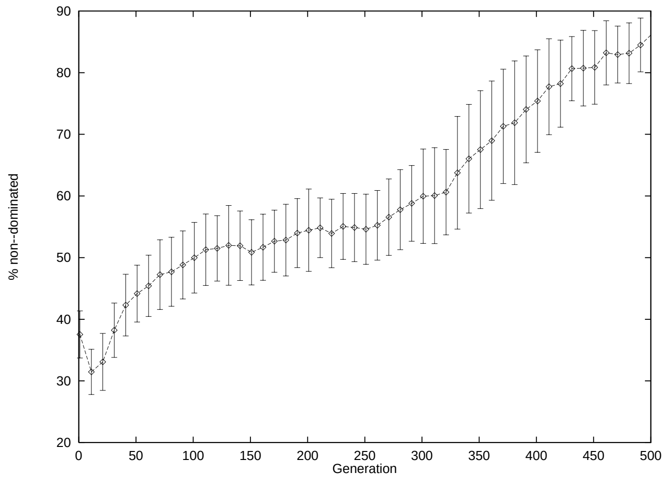

Figure 2.2: Non–dominated percent of the population for tournament selection of size 2, average over 50 runs. Average with standard deviation as error bars.

2.3.1 *Pareto ranking*

Instead of just a dominant/non-dominant scheme for Pareto sorting that doesn't distinguish elements very well (as we can see from the Figure 2.2 where at the end of the run more then 80% are non–dominated) and because of the arguments presented above, we can use a finer method: Pareto ranking (compare [12, 13]).

Definition 5 *Pareto rank* r in a set $\mathbf{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ is assigned in the following way:

$(\forall \boldsymbol{x} \in \mathbf{X}) r(\boldsymbol{x}) \leftarrow 1$
$(\forall \boldsymbol{x} \in \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\})(\forall \boldsymbol{y} \in \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \setminus \{\boldsymbol{x}\})$ // sequentially!
If $(r(\boldsymbol{x}) = r(\boldsymbol{y}) \wedge \boldsymbol{x} > \boldsymbol{y})$ $r(\boldsymbol{x}) \leftarrow r(\boldsymbol{x}) + 1$
If $(r(\boldsymbol{x}) = r(\boldsymbol{y}) \wedge \boldsymbol{x} < \boldsymbol{y})$ $r(\boldsymbol{y}) \leftarrow r(\boldsymbol{y}) + 1$

Using the above definition/algorithm, points A and D from Figure 2.1 would have rank 6, B and C rank 5 and point E rank 2.

2.4 Optimisation results

Figure 2.3 shows the Pareto front obtained by performing Pareto optimisation on the BAe function optimising y_3, y_4 and y_9.

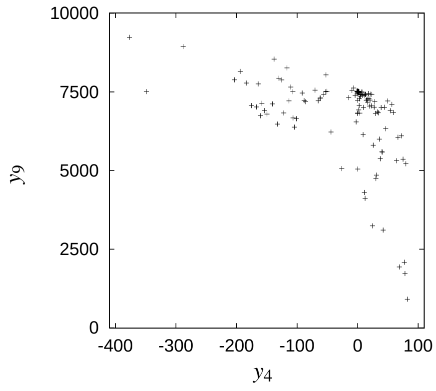

Figure 2.3: Pareto front of y_4 versus y_9 for BAe function.

2.4.1 *What to optimise?*

We have 9 output parameters to optimise, some of them are very complementary. For instance, if we want to optimise on y_3, y_4 and y_9, we have the situation[1] as presented in Table 2.2 (one run on GA with Pareto sorting).

Table 2.2 shows that optimising y_4 (specific excess power (SEP) for supersonic case) affects ferry range (y_9) a great deal and vice versa. If we want to maximise ferry range, we obtain catastrophic values for SEP, if we maximise SEP, we get ferry range 0. Since the goal of the project is to design of an aircraft that performs well in all situations, we must consider both sub-sonic and supersonic configurations.

If we use weights and the method of positive ideal solution for assigning weights, we get the results shown in Table 2.3. Because of the interaction between objectives,

[1] If we optimise on y, we use additional constraint $y \geq 0.001$

Optimise	y_3	y_4	y_9
y_3	148.1757	76.9852	0.0000
y_4	145.2489	82.9057	0.0000
y_9	-1.7305	-492.6476	10263.6426
y_3, y_4	145.3017	82.8891	2310.7244
y_3, y_9	23.2492	-380.7826	10096.2207
y_4, y_9	115.2269	0.0001	7886.8315
y_3, y_4, y_9	115.7906	0.0547	7873.7720

Table 2.2: Optimising different combinations of objectives gives different results

we have put additional constraints that every objective that we optimise on must be greater then 0.001.

Those results clearly show that choosing different optimisation methods and different objectives can give totally different results that are, globally, not necessary better or worse, just different.

What?	How?	y_3	$\sigma(y_3)$	y_4	$\sigma(y_4)$	y_9	$\sigma(y_9)$
y_3, y_4, y_9	PAR	119.72	4.27	0.68	3.65	7747.15	134.69
	LEX	115.61	5.17	0.42	1.33	7826.03	100.81
	WEI	146.84	ε	76.93	ε	6070.39	0.64
y_3, y_9	PAR	5.62	6.27	-459.52	28.27	10220.1	43.41
	LEX	0.34	0.64	-482.91	2.72	10249.8	10.47
	WEI	139.56	0.08	52.26	0.19	6959.5	5.93
y_4, y_9	PAR	117.33	4.41	0.29	1.19	7811.8	106.24
	LEX	113.68	5.35	0.17	0.80	7846.14	84.12
	WEI	145.72	ε	76.16	0.002	6198.94	0.29
y_3, y_4	PAR	145.30	ε	82.89	ε	1838.35	1734.38
	LEX	145.30	ε	82.89	ε	1527.58	1641.24
	WEI	145.29	ε	82.90	ε	1637.69	1686.49
y_9	PAR	-1.74	0.02	-491.83	1.25	10262.8	0.96
	LEX	-1.74	0.01	-491.98	0.79	10263.4	0.45
	WEI	-1.74	0.01	-492.19	0.81	10263.4	0.31
$y_1, \ldots, y_9$	PAR	59.67	ε	9.81	ε	4787.75	0.02
	LEX	59.67	ε	9.81	ε	4787.73	0.03
	WEI	114.66	0.16	37.19	0.21	5769.41	15.22

Table 2.3: Pareto (PAR), random lexicographic (LEX) and weighted sum (WEI) optimisations, average over 50 runs. Here $0 < \varepsilon < 0.005$ and $\sigma(y)$ is the standard deviation of y.

2.5 VEGA

VEGA stands for Vector Evaluated Genetic Algorithm and it was developed by Schaffer [14, 3]. The basic algorithm is presented in Figure 2.4.

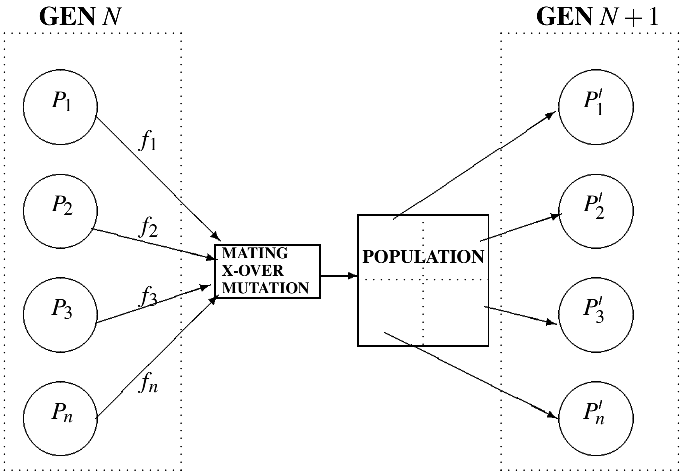

Figure 2.4: Schema of Schaffer's (basic) VEGA algorithm

The whole population P is divided into several (equal to the number of objectives we are optimising) sub–populations. Operators used are the same as in a standard GA, only the selection step is different: from each subpopulation (of size m) choose individuals for the mating pool taking into account only that one objective. Then we perform mating, crossover and mutation in the standard way and split the new obtained population into sub–populations randomly. More formally:

Let $\mathcal{D}$ be a domain and $\mathbf{F}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x}))$ the function with domain $\mathcal{D}$. We define the following operators:

$\mathrm{CROSS} : \mathcal{D}^2 \mapsto \mathcal{D}$	– crossover
$\mathrm{MUT} : \mathcal{D} \mapsto \mathcal{D}$	– mutation
$\mathrm{SEL}_f : \mathcal{D}^m \mapsto \mathcal{D}^m$	– selection according to objective f

Let $P(t) = (P_1, \ldots, P_k)$ be the population in generation t. Then

$$P(t+1) = \mathrm{MUT}\big(\mathrm{CROSS}(\cup_{i=1}^{k} \mathrm{SEL}_{f_i}(P_i))^2\big) \tag{4}$$

Later analysis of the algorithm, shows that a problem with VEGA is that it tends to average the solution. According to [15, p.191] (which further refers to [16]) "analysis of VEGA shows that the effect is the same as if fitness were a linear combination of attributes". Several approaches have been proposed to eliminate this disadvantage [17].

A similar approach to VEGA has been developed by Fourman [18]: selection is performed by comparing pairs of individuals, each pair according to one objective selected at random.

Actually, concerning Schaffer's VEGA, there are additional features that one can introduce with it. In the original, there is no real concept of subpopulations: we shuffle the populations together in every generation. So, the basic three ideas are

- Wait t_s generations before mixing them together;
- Don't mix them together, but every t_m generations copy n_b individuals from populations to other. In that case, if k is the number of subpopulations (i.e. number of objectives) and N population size (so, N/k is the size of each subpopulation), we have constraint $n_b \leq N/k^2$.
- Similar to the previous idea, but don't copy, migrate individuals.

2.5.1 *Results*

We have implemented those ideas and some results and a comparison with the Pareto method are presented in Table 2.4. All results are averaged over 200 runs. All GA parameters are the same in all runs except for population size in the case of Schaffer's VEGA. We used uniform crossover with probability 1 and exponential mutation with probability 1/8 applied on a real–coded chromosomes.

method	y_3	$\sigma(y_3)$	y_4	$\sigma(y_4)$	y_9	$\sigma(y_9)$	$\sum y_i$
Fourman	115.30	8.37	12.23	10.90	7593.0	143.6	7720.48
Pareto	120.07	4.57	0.44	2.41	7731.7	145.3	7852.17
Schaffer	145.67	4.55	71.8	13.0	5886.4	230.3	6103.93
Subpop (mix)	116.76	8.90	20.7	13.6	7355.5	159.8	7492.96
Subpop (copy)	114.94	11.14	20.3	16.0	7116.1	260.0	7251.32

Table 2.4: Results on maximising y_3, y_4 and y_9 using different optimisation methods.

3 Conclusion and Future Work

In this paper we have analysed several different methods for optimising multi–objective functions. The results clearly show that there is a different problem here that we do not have in the case of single objective functions: the problem of relative importance and interaction of objectives. That also makes a problem of evaluating solutions much more difficult as it is not simple to say: this one is the best solution!

All optimisation methods presented above need a quantitative value for the relative importance of objectives and thus present an additional difficult problem. So, the requirement is for methods that can work with a qualitative in addition to a quantitative characterisation of importance ('this one is definitely more important than the other', 'this one is not so important' etc.). Preference methods [19, 20], as well as various fuzzy set methods [21, 22, 23] and combinations thereof [24] should be included here.

Also, human interaction should not be forgotten. Methods for exploring, expanding and sampling the search space in all directions are needed. In the future work it is intended to introduce some agent based systems which model, to some extent, the human–based solution evaluation process.

References

[1] Ching-Lai Hwang and Abu Syed Md. Masud. *Multiple Objective Decision Making – Methods and Applications*. Springer Verlag, Berlin, 1979.

[2] Andrzej Osyczka. *Multicriterion Optimization in Engineering with FORTRAN Programs*. Ellis Horwood, 1984.

[3] J. David Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In Grefenstette [25], pages 93–100.

[4] Jeffrey Horn and Nicholas Nafpliotis. Multiobjective optimization using the niched Pareto genetic algorithm. Technical Report IlliGAL Report No. 93005, Illinois Genetic Algorithm Laboratory, 1993.

[5] Carlos M. Fonseca and Peter J. Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–16, 1995.

[6] Mitsuo Gen and Runwei Cheng. *Genetic Algorithms & Engineering Design*. Wiley Series in Engineering Design and Automation. J. Wiley & Sons, 1997.

[7] Carlos M. Fonseca and Peter J. Fleming. Multiple objective optimization and multiple constraint handling with evolutionary algorithms I: A unified formulation. Technical Report 564, University of Sheffield, UK, January 1995.

[8] Ian C. Parmee and Mike J. Denham. The integration of adaptive search techniques with current engineering design practice. In Ian C. Parmee, editor, *Adaptive Computing in Engineering Design and Control '94*, pages 1–14. Plymouth Engineering Design Centre, 1994.

[9] Ian C. Parmee. Strategies for the integration of evolutionary/adaptive search with the engineering design process. In D. Dasgupta and Z. Michalewicz, editors, *Evolutionary Algorithms in Engineering Applications*, pages 453–477. Springer Verlag, 1997.

[10] Eric Webb. MINICAPS – a simplified version of CAPS for use as a research tool. Unclassified Report BAe-WOA-RP-GEN-11313, British Aerospace, July 1997.

[11] Aharon Ben-Tal. Characterisation of Pareto and lexicographic optimal solutions. In Fandel and Gal [20], pages 1–11.

[12] N. Srinivas and Kalyanmoy Deb. Multiobjective optimization using nondominant sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1995.

[13] Manuel Valenzuela-Rendón and Eduardo Uresti-Chare. A non–generational genetic algorithm for multiobjective optimization. In Thomas Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 658–665. Morgan Kaufmann, July 1997.

[14] J. David Schaffer. *Some Experiments in Machine Learning using Vector Evaluated Genetic Algorithm*. PhD thesis, Vanderbilt University, Nashville, 1984. TCGA file No. 00314.

[15] Jon T. Richardson, Mark R. Palmer, Gunar Liepins, and Mike Hilliard. Some guidelines for genetic algorithms with penalty functions. In David Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithm*, pages 191–197, 1989.

[16] M. Hilliard, G. Liepins, M. Palmer, and G. Rangarajan. The computer as a partner in algorithmic design: Automated discovery of parameters for multi-objective heuristics. In *Conference on the Impact of Recent Computer Advances in Operations Research*, 1988.

[17] Ian C. Parmee, M. Johnson, and S. Burt. Techniques to aid global search in engineering design. In Frank D. Anger, Rita V. Rodrigez, and Moonis Ali, editors, *Industrial and Engineering Applications of Artificial Intelligence and Expert Systems IAE/AIE '94*, Austin, Texas, 1994. Gordon and Breach Science Publishers.

[18] M. P. Fourman. Compaction of symbolic layout using genetic algorithms. In Grefenstette [25], pages 141–153.

[19] Dragan Cvetković. The logic of preference and decision supporting systems. Master's thesis, Faculty of Mathematics, University of Belgrade, 1993. Also Technical Report MPI-I-93-260 of Max Planck Institute for Computer Science, Saarbrücken, Germany.

[20] G. Fandel and T. Gal, editors. *Proceedings of the Third Conference on Multiple Criteria Decision Making Theory and Application*, number 177 in Lecture Notes in Economics and Mathematical Systems, Hagen/Königswinter, West Germany, August 1979. Springer Verlag.

[21] Fabrice Wawek, Anne-Marie Jolly-Desodt, and Daniel Jolly. Modelisation of the criterias coming from the human operator for a fuzzy decision support system. In *4th European Congress on Intelligent Techniques and Soft Computing EUFIT '96*, pages 1349–1352, Aachen, 1996.

[22] Lofti A. Zadeh. The concept of linguistic variable and its application to approximate reasoning. *Information Sciences*, 1975.

[23] Tamás Zétényi, editor. *Fuzzy Sets in Psychology*. Number 56 in Advances in Psychology. North–Holland, 1988.

[24] János Fodor and Marc Roubens. *Fuzzy Preference Modelling and Multicriteria Decision Support*. Kluwer Academic Publishers, 1994.

[25] John J. Grefenstette, editor. *Proceedings of the First International Conference on Genetic Algorithms*. Lawrence Erlbaum Associates, 1985.