Chapter 7
Multi-objective Optimization Using Surrogates

Ivan Voutchkov and Andy Keane

Abstract. Until recently, optimization was regarded as a discipline of rather theoretical interest, with limited real-life applicability due to the computational or experimental expense involved. Practical multiobjective optimization was considered almost as an utopia even in academic studies due to the multiplication of this expense. This paper discusses the idea of using surrogate models for multiobjective optimization. With recent advances in grid and parallel computing more companies are buying inexpensive computing clusters that can work in parallel. This allows, for example, efficient fusion of surrogates and finite element models into a multiobjective optimization cycle. The research presented here demonstrates this idea using several response surface methods on a pre-selected set of test functions. We aim to show that there are number of techniques which can be used to tackle difficult problems and we also demonstrate that a careful choice of response surface methods is important when carrying out surrogate assisted multiobjective search.

7.1 Introduction

In the world of real engineering design, there are often multiple targets which manufacturers are trying to achieve. For instance in the aerospace industry, a general problem is to minimize weight, cost and fuel consumption while keeping performance and safety at a maximum. Each of these targets might be easy to achieve individually. An airplane made of balsa wood would be very light and will have low fuel consumption, however it will not be structurally strong enough to perform at high speeds or carry useful payload. Also such an airplane might not be very safe,

Ivan Voutchkov
University of Southampton, Southampton SO17 1BJ, United Kingdom
e-mail: iiv@soton.ac.uk

Andy Keane
University of Southampton, Southampton SO17 1BJ, United Kingdom
e-mail: ajk@soton.ac.uk

i.e., robust to various weather and operational conditions. On the other hand, a solid body and a very powerful engine will make the aircraft structurally sound and able to fly at high speeds, but its cost and fuel consumption will increase enormously. So engineers are continuously making trade-offs and producing designs that will satisfy as many requirements as possible, while industrial, commercial and ecological standards are at the same time getting ever tighter.

Multiobjective optimization (MO) is a tool that aids engineers in choosing the best design in a world where many targets need to be satisfied. Unlike conventional optimization, MO will not produce a single solution, but rather a set of solutions, most commonly referred to as Pareto front (PF) [12]. By definition it will contain only non-dominated solutions[1]. It is up to the engineer to select a final design by examining this front.

Over the past few decades with the rapid growth of computational power, the focus in optimization algorithms in general has shifted from local approaches that find the optimal value with the minimal number of function evaluations to more global strategies which are not necessarily as efficient as local searches but (some more than the others) promise to converge to global solutions, the main players being various strands of genetic and evolutionary algorithms. At the same time, computing power has essentially stopped growing in terms of flops per CPU core. Instead parallel processing is an integral part of any modern computer system. Computing clusters are ever more accessible through various techniques and interfaces such as multi-threading, multi-core, Windows HPC, Condor, Globus, etc.

Parallel processing means that several function evaluations can be obtained at the same time, which perfectly suits the ideology behind genetic and evolutionary algorithms. For example Genetic algorithms are based on the idea borrowed from biological reproduction, where the offspring of two parents copy the best genes of their parents but also introduce some mutation to allow diversity. The entire generation of offspring produced by parents in a generation represent designs that can be evaluated in parallel. The fittest individuals survive and are copied into the next generation, whilst weak designs are given some random chance with low probability to survive. Such parallel search methods are conveniently applicable to multiobjective optimization problems, where the fitness of an individual is measured by how close to the Pareto front this designs is. All individuals are ranked, those that are part of the Pareto front get the lowest (best) rank, the next best have higher rank and so on. Thus the multiobjective optimization is reduced to single objective minimization of the rank of the individuals. This is idea has been developed by Deb and implemented in NSGA2 [5].

In the context of this paper, the aim of MO is to produce a well spread out set of optimal designs, with as few function evaluations as possible. There are number of methods published and widely used to do this – MOGA, SPEA, PAES, VEGA, NSGA2, etc. Some are better than others - generally the most popular in the literature are NSGA2 (Deb) and SPEA2 (Zitzler), because they are found to achieve good results for most problems [2, 3, 4, 5, 6]. The first is based on genetic algorithms and

[1] Non-dominated designs are those where to improve performance in any particular goal performance in at least one other goal must be made worse.

the second on an evolutionary algorithm, both of which are known to need many function evaluations. In real engineering problems the cost of evaluating a design is probably the biggest obstacle that prevents extensive use of optimization procedures. In the multiobjective world, this cost is multiplied, because there are multiple expensive results to obtain. Evaluating directly a finite element model can take several days, which makes it very expensive to try hundreds or thousands of designs.

7.2 Surrogate Models for Optimization

It seems that increased computing power leads to increased hunger for even more computing power, as engineers realise that they can run more detailed and realistic models. In essence, from an engineering point of view, the available computing power is never enough and this tendency does not seem to be changing at least in the foreseeable future. To put these words into prospective, to be useful to an engineering company, a modern optimization approach should be able to tackle a global multiobjective optimization problem in about a week. The problem would typically have 20-30 variables, 2-5 objectives, 2-5 constraints with evaluation times of about 12-48h per design and often per objective. Unless you have access to 5000-7000 parallel CPUs, the only way to currently tackle such problems is to use surrogate models.

In the single objective world, approaches using surrogate models are fairly well established and have proven to successfully deal with the problem of computational

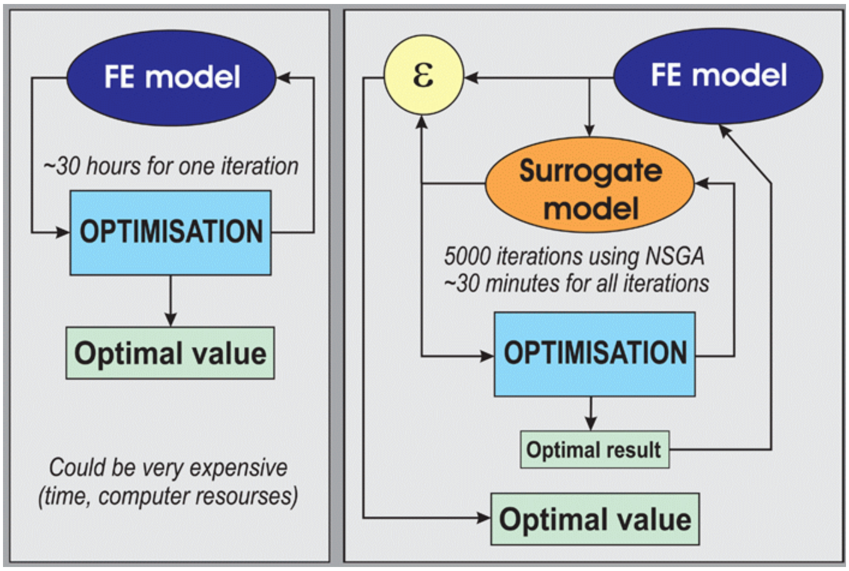
FE model
~30 hours for one iteration
OPTIMISATION
Optimal value
Could be very expensive (time, computer resourses)
ε
FE model
Surrogate model
5000 iterations using NSGA
~30 minutes for all iterations
OPTIMISATION
Optimal result
Optimal value

Fig. 7.1 Direct search versus surrogate models for optimization

expense (see Fig. 7.1) [22]. Since their introduction, more and more companies have adopted surrogate assisted optimization techniques and some are making steps to incorporate this approach in their design cycle as standard. The reason for this is that instead of using the expensive computational models during the optimization step, they are substituted with a much cheaper but still accurate replica. This makes optimization not only useful, but ***usable and affordable***. The key idea that makes surrogate models efficient is that they should become more accurate in the region of interest as the search progresses, rather than being equally accurate over the entire design space, as an FE representation will tend to be. This is achieved by adding to the surrogate knowledge base only at points of interest. The procedure is referred to as ***surrogate update***.

Various publications address the idea of surrogates models and multiobjective optimisation [10, 11, 12, 13, 14, 15, 16, 17, 18, 19]. As one would expect, no approximation method is universal. Factors such as function modality, number of variables, number of objectives, constraints, computation time, etc., all have to be taken into account when choosing an approximation method. The work presented here aims to demonstrate this diversity and hints at some possible strategies to make best use of surrogates for multi-objective problems.

7.3 Multi-objective Optimization Using Surrogates

To illustrate the basic idea, the zdt1 – zdt6 test function suite [3] will be used to begin with. It is a good suite to demonstrate the effectiveness of surrogate models, as it is fairly simple for response surface (surrogate) modelling. Fig. 13.3 represents the zdt2 function and the optimisation procedure. It is a striking comparison, demonstrating the surrogate approach. The problem has two objective functions and two design variables. The Pareto front obtained using surrogates with 40 function evaluations is far superior to the one without surrogates and the same number of function evaluations.

Table 7.1 Full function evaluations for ZDT2 - Fig. 13.3

Number of variables	2	5	10
Number of function evaluations without surrogates	2500	5000	10000
Number of function evaluations with surrogates	40	40	60

On the other hand 2500 evaluations without surrogates were required to obtain a similar quality of Pareto front to the case with surrogates and 40 evaluations. The difference is even more significant if more variables are added – see Table 14.1.

Here we have chosen objective functions with simple shapes to demonstrate the effectiveness of using surrogates. Both functions would be readily approximated using most available methods. It is not uncommon to have relationships of similar simplicity in real problems, although external noise factors could make them

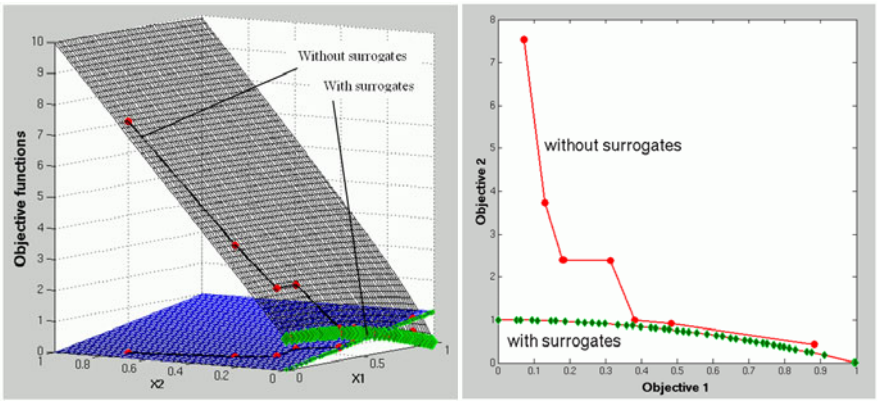

Fig. 7.2 A (left) – Function ZDT2; **B (right) –** ZDT2 – Pareto front achieved in 40 evaluations: Diamonds – Pareto front with surrogates; Circles – solution without surrogates

look rougher. Relationships of higher order of multimodality would be more of a challenge for most methods, as will be demonstrated later.

7.4 Pareto Fronts - Challenges

Depending on the search algorithm, the quality of the Pareto front could vary greatly. There are various characteristics that describe a good quality Pareto front:

1. Spacing – better search techniques will space the points on the Pareto front uniformly rather than producing clusters. See Fig. 7.3a
2. Richness – better search techniques will put more points on the Pareto front than others. See Fig. 7.3b
3. Diversity – better search techniques will produce fronts that are spread out better with respect to all objectives. See Fig. 7.3c
4. Optimality – better search techniques will produce fronts that dominate the fronts produced by less good techniques. In test problems this is usually measured as 'generational distance' to an ideal Pareto front. We discuss this later. See Fig. 7.3d
5. Globality – the obtained Pareto front is a global as opposed to local. Similar to single objective optimization, in the multiobjective world, it is also possible to have local and global optimal solutions. This concept is demonstrated using the F5 test function (a full description is given in sections 15.4 and 15.5). Fig. 7.4 illustrates the function and the optimization procedure. Due to the sharp nature of the global solution it cannot be guaranteed that with a small number of GA evaluations, the correct solution will be found. Furthermore, since the surrogate is based only on sampled data, if this data does not contain any points in the global optimum area, then the surrogate will never know about its existence. Therefore

any optimization based only on such surrogates will lead us to the local solution. Therefore conventional optimization approaches based on surrogate models rely on constant updating of the surrogate. A widely accepted technique in single objective optimization is to update the surrogate with its current optimal solution. In multiobjective terms this will translate to updating the surrogate with one or more points belonging to its Pareto front. If the surrogate Pareto front is local and not global, then the next update will also be around the local Pareto front. Continuing with this procedure the surrogate model will become more and more accurate in the area of the local optimal solution, but will never know about the existence of the global solution.

6. Robust convergence from any start design with any random number sequence. It turns out that the success of a conventional multiobjective optimization based on surrogates, using updates at previously found optimal locations strongly depends on the initial data used to train the first surrogate before any updates are added. If this data happens to contain points around the global Pareto front, then the algorithm will be able to quickly converge and find a nice global Pareto front. However the odds are that the local Pareto fronts are smoother and easier to find shapes and in most cases this is where the procedure will converge unless suitable global exploration steps are taken.
7. Efficiency and convergence – better search techniques will converge using less function evaluations.

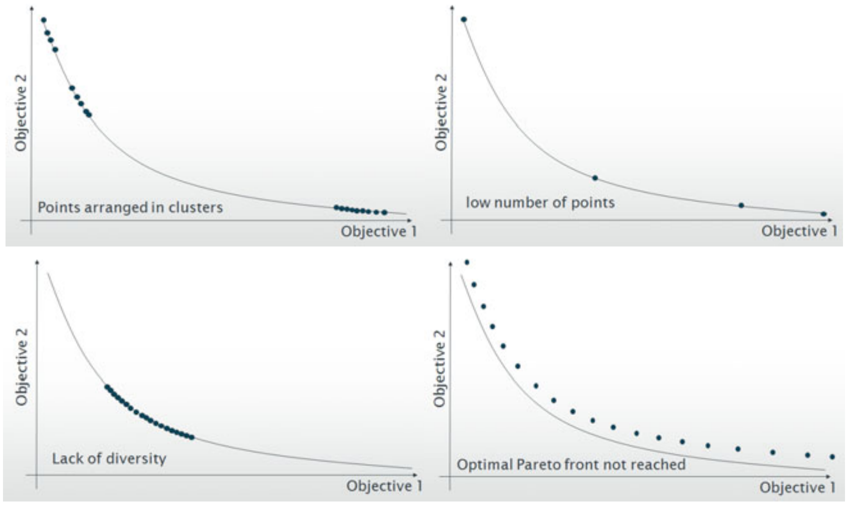

Fig. 7.3 Pareto front potential problems - **(a)** clustering; (b) too few points; (c) lack of diversity; (d) non-optimality

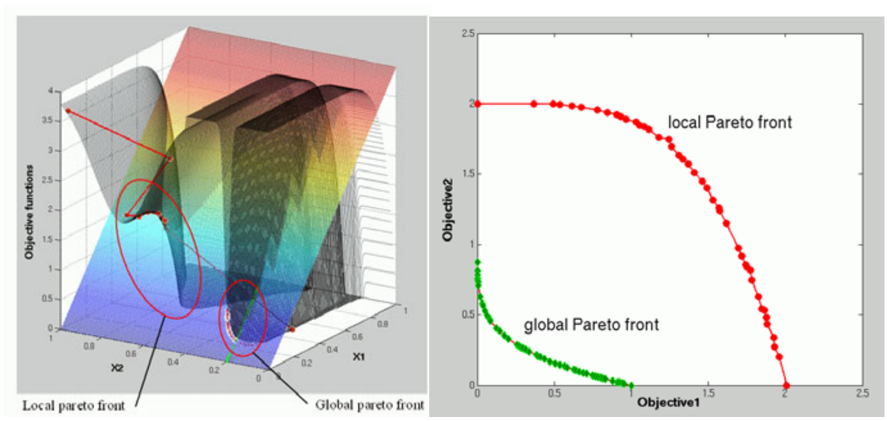

Fig. 7.4 F5: Local and global solutions

7.5 Response Surface Methods, Optimization Procedure and Test Functions

In a previous publication [20] we have shown that for complex and high-dimensional functions Kriging is the response surface method of choice [22]. We have also stressed the importance of applying a high level of understanding when using Kriging. There have been various publications that critique kriging, due to lack of understanding. Our opinion is that if the user understands the strengths and weaknesses of this approach it can become an invaluable tool, often the only one capable of producing meaningful results in reasonable times.

Kriging is a Response Surface (RSM) method, designed in the 60's for geological surveys [7]. It can be a very efficient RSM model for cases where it is expensive to obtain large amounts of data. A significant number of publications discuss the kriging procedure in detail. An important role for the success of the method is the tuning of its hyper parameters. It should be mentioned that researchers who have chosen rigorous training procedures, report positive results when using kriging, while publications that use basic training procedures often reject this method. Nevertheless, the method is becoming increasingly popular in the world of optimization as it often provides a surrogate with usable accuracy.

This method was used to build surrogates for the above test cases, therefore it is useful to briefly outline its major pros and cons:

Pros:

- can always predict with no error at sample points,
- the error in close proximity to sample points is minimal,
- requires small number of sample points in comparison to other response surface methods,
- reasonably good behaviour with high dimensional problems.

Cons:

- for large number of data points and variables, training of the hyper-parameters and prediction may become computationally expensive.

Researchers should make a conscious decision when choosing Kriging for their RSMs. Such a decision should take into account the cost of a direct function evaluation including constraints (if any), available computational power, and dimensionality of the problem. Sometimes it might be possible to use kriging for one of the objectives while another is evaluated directly, or a different RSM is used to minimize the cost.

As this paper aims to demonstrate various approaches in making a better use of surrogate models, we will use kriging throughout, but most conclusions could be generalised for other RS methods as well. The chosen multiobjective algorithm is NSGA2. Other multiobjective optimizers might show slightly different behaviour. The basic procedure is as follows:

1. Carry out 20 LPtau [8] spaced initial direct function evaluations.
2. Train hyper-parameters, using combination of GA and DHC (dynamic hill climbing) [23]
3. Choose a selection of update strategies with specified number of updates.
4. Search the RSMs using each of the selected methods
5. Select designs that are best in terms of ranking and space filling properties.
6. Evaluate selected designs and add to data set.
7. Produce Pareto front and compare with previous. Stop if 2-3 consecutive Pareto fronts are identical. Otherwise continue.
8. If Pareto front contains too many points, choose specified number of points that are furthest away from each other
9. Repeat from step 2.

There are several possible stopping criteria:

- fixed number of update iterations,
- stop when all update points are dominated,
- stop if the percentage of new update points that belong to the Pareto front falls below a pre-defined value,
- stop if the percentage of old points on the current Pareto front rises above a pre-defined value,
- stop when there is no further improvement of the Pareto front quality. The quality of the Pareto front is a complex multiobjective problem on its own. The best Pareto front could be defined as the one being as close as possible to the origin of the objective function space, while having the best diversity, i.e., spread on all

objectives and the points are evenly distributed. Metrics for assessing the quality of the Pareto front are discussed by Deb [3].

We have used the last of these criteria for our studies.

7.6 Update Strategies and Related Parameters

One of the main aims of this publication is to show the effect of different update strategies and number of updates. Here we consider the following six approaches in various combinations:

- UPDMOD = 1; (*Nr*) - Random updates. These can help escape from local Pareto fronts and enrich the genetic material,
- UPDMOD = 2; (*Nrsm*) - RSM Pareto front. A specified number of points are extracted from the Pareto front obtained after the search of the response surface models of the objectives and constraints (if any). When the RSM Pareto front is rich it is possible to extract data that is uniformly distributed.
- UPDMOD = 3; (*Nsl*) - Secondary NSGA2 layer. A completely independent NSGA2 algorithm is applied directly to the non-RSM objective functions and constraints. This exploits the well known property of the NSGA2 which makes it (slowly) converge to global solutions. During each update iteration, the direct NSGA2 is run for one generation with population size of *Nsl*. There are two strands to this approach. The first one is referred to as ‘decoupled’. The genetic material is completely independent from the other update strategies. No entries other than those from the direct NSGA2 are used. The second strand is referred to as ‘coupled’, where the genetic information is composed of suitable designs obtained by other participating update strategies. Suitable designs are selected in terms of Pareto optimality, or rank in terms of NSGA2. Please note that although it might sound similar, this is a completely different approach from the MμGA algorithm, proposed by Coello and Toscano (2000)
- UPDMOD = 4; (*Nrmse*) – Root of the Mean Squared Error (RMSE). When using kriging as a response surface model, it is possible to compute an estimate of the RMSE, at no significant computational cost. The value of this metric is large where there are large gaps between data points. RMSE is minimal close to or at existing data points. Therefore adding updates at the location of the maximum RMSE should significantly improve the quality and coverage of the response surface model. When dealing with multiple objectives/constraints it is appropriate to construct a Pareto front of maximum RMSEs for all objectives and extract *Nrmse* points from it.
- UPDMOD = 5; (*Nie*) – Expected improvement (EI). This is another kriging specific function which represents the probability of finding the optimal point in a new location. The update points are extracted from the Pareto front of the maximum values of the EI for all objectives. For constrained problems, the values of EI for all objectives are multiplied by the value of the feasibility of the constraints, which is 1 for satisfied constraints 0 for unfeasible and rather smooth ridge around the constraint boundary, see Forrester *et al* [22].

- UPDMOD = 6; (*Nkmean*) – The RSMs are searched using GA or DHC and points are extracted using a *k-mean* cluster detection algorithm.

All these update strategies have their own strengths and weaknesses, and therefore a suitable combination should be carefully considered. The results section of this chapter provides some insights on the effects of each of these strategies when used in various combinations.

Additional Parameters that Can Affect the Search
The following parameters can also affect the performance of a multi-objective RSM search:

- RSMSIZE – number of points used for RSM construction. It is expected that the more points that are used, the more accurate the RSM predictions, however this comes at increasing training cost. Therefore the number of training points should be limited.
- EVALSIZE – number of points used during RSM evaluation. This stage is considerably less expensive than training and therefore more points can be used during the evaluation stage. Ultimately this should increase the density of quality material and therefore fewer gaps for the RSM to approximate.
- EPREWARD – endpoint reward factor. Higher value rewards are given at the end points of the Pareto front, and this improves its spread. Lower value would increase the pressure of the GA to explore the centre of the Pareto front.
- GA_NPOP and GA_NGEN – the population size and number of generation used to search the RSM, RMSE and EI Pareto fronts.

7.7 Test Functions

Several test functions with various degrees of complexity have been chosen to demonstrate the overview of the RS methods for the purpose of multiobjective optimization. These functions are well known from the literature:

F5: (Fig. 7.4). High complexity shape – has a smooth and a sharp feature. The combination of both makes it easier for the optimization procedure to converge to the smooth feature, which represents a local Pareto front. The global Pareto front lies around the sharp feature which is harder to reach. Two objectives, $x\,(i) = 0 \,..\, 1$, $i = 1, 2$; no constraints [3], page 350.

ZDT1 - ZDT6: Clustered and discontinuous Pareto fronts. Shape complexity is moderate. Two objectives, n variables (in present study $n = 2$), no constraints. $x\,(i) = 0 \,..\, 1$, $i = 1, 2$ [3], page 357.

ZDT1cons: Same formulation as for ZDT1 but with 25 variables and 2 constraints. Constraints are described in [3], page 368.

Bump: The bump function, 25 variables, 2 objectives, 1 constraint. We have used the function as provided in [21] which is a single objective with two constraints. We have made one of the constraints into second objective, so that the optimization problem is defined as : Maximise the original objective, minimize the sum of variables whilst keeping the product of the variables greater than 0.75. There are 25 variables, each varying between 0 and 3.

7.8 Pareto Front Metrics

To measure the performance of the various strategies discussed in this paper, we have adopted several metrics. Some of them use comparison to an 'ideal' solution which is denoted by Q and represents the Pareto front obtained using direct search with a large number of iterations (20,000). All metrics are designed so that smaller is better.

7.8.1 Generational Distance ([3], pp.326)

The average of the minimum Euclidian distance between each point of the two Pareto fronts,

$$gd = \frac{\sqrt{\sum_{i=1}^{|Q|} d_i}}{|Q|},$$

$d_i = \min_{k=1,|p|} \sqrt{\sum_{j=1}^{M} \left(f_j^{(i)} - p_j^{(k)}\right)^2}$, and is the Euclidian distance between the solution (i) and the nearest member of Q.

7.8.2 Spacing

Standard deviation of the absolute differences between the solution (i) and the nearest member of Q,

$$sp = \sqrt{\frac{1}{|Q|} \sum_{i=1}^{|Q|} \left(d_i - \bar{d}\right)^2},$$

$$d_i = \min_{k=1,|p|} \sum_{j=1}^{M} \left(f_j^{(i)} - p_j^{(k)}\right).$$

7.8.3 Spread

$$\Delta = 1 - \frac{\sum_{m=1}^{M} d_m^e - \sum_{i=1}^{|Q|} \left|d_i - \bar{d}\right|}{\sum_{m=1}^{M} d_m^e + |Q|\bar{d}},$$

where d_i is the absolute difference between neighbouring solutions. For compatibility with the above metrics, the values of the spread is subtracted from 1, so that a wider spread will produce a smaller value.

7.8.4 Maximum Spread

Normalized distance between the most distant points on the pareto front. The distance is normalized against the maximum spread of the 'ideal' pareto front. For compatibility with the above metrics, the value of the maximum spread is subtracted from 1, so that a wider spread will produce a smaller value,

$$MS = 1 - \sqrt{\frac{1}{M}\sum_{m=1}^{M}\left(\frac{\max\limits_{i=1,|Q|} f_m^{(i)} - \min\limits_{i=1,|Q|} f_m^{(i)}}{P_m^{\max} - P_m^{\min}}\right)^2}\,.$$

7.9 Results

The study carried out aims to show the effect of applying various update strategies, number of training and evaluation points, etc. The performance of each particular approach is measured using the metrics described in the previous section.

An overall summary is given at the end of this section, but the best recipe appears to be highly problem dependant. It is also not possible to show all results for all functions due to limited space, and we have therefore chosen several that best represent the ideas discussed.

To correctly appreciate the results, please bear in mind that they are meant to show diversity rather than a magic recipe that works in all situations.

7.9.1 Understanding the Results

The legend on the figures represents the selected strategy in the form

[*Nr*]-[*Nrsm*]-[*Nsl*]-[*Nrmse*]-[*Nie*]-[*Nkmean*]MUPD[*RSMSIZE*]MEVL[*EVALSIZE*]

so that a 8-14-15-10-3-3MUPD50MEVL300 would represent 8 random update points, 14 RSM updates, 15 NSGA2 Second layer updates, 10 RMSE updates, 3 EI updates, 3 KMEAN updates with 50 krig training points and 300 krig evaluation points.

All approaches were given a maximum of 60 update iterations and stopping criteria of reaching two consecutive unchanged Pareto fronts. Total number of runs is recorded for each update iteration and all metrics are plotted against number of real function evaluations, (i.e. likely cost on real expensive problems).

Strategies with 'dec' appended to their name – indicate that the decoupled Second layer is used, as opposed to coupled for those where *Nsl* = 30 and without any appendix. Those labled '43' use a one pass constraint penalty expected improvement strategy whilst those that have *Nie* = 30 and no appendix use a constraint feasibility algorithm.

7.9.2 Preliminary Calculations

7.9.2.1 Finding the Ideal Pareto Front

As mentioned in section 7.8, most of the Pareto front metrics are based on comparison to an ‘ideal’ Pareto front. To find it, each of the test functions has been run through a direct NSGA2 search (direct = without the usage of surrogates) with Population size of 100 for 200 generations, which takes 20000 function evaluations.

7.9.2.2 How Many Generations for the RSM Search?

We have conducted a study for each of the test functions to find what the minimum number of generations they should be run for is, in order to achieve best convergence. We found that a population size of 70 with 80 generations is sufficient for all of test problems and this is what we have used for our tests. Some test functions, such as ZDT1 - ZDT6 with two variables could be converged using a smaller number of individuals and generations, however for comparison purposes we decided to use the same settings for all functions.

7.9.2.3 What Is the Best Value for EPREWARD during the RSM Search?

The EPREWARD value is strictly individual for each function. Taking into account the specifics of the test function it can improve the diversity of the Pareto front. The default value is 0.65, which works well for most of the functions, but we have also conducted studies where this parameter is varied between -1 and 1 in steps of 0.1, and individual value for each function is selected based on best Pareto front metrics.

7.9.3 The Effect of the Update Strategy Selection

Fig. 7.5 shows that the selection of update strategy is important even for functions with only two variables. F5 has a deceptive Pareto front and several update strategies were not able to escape from the local Pareto front.

Fig. 7.6 clearly shows that some strategies have converged earlier than the others, but some of them to the local front. Generally methods such as Random updates and Secondary NSGA2 layer updates are not based on the RSM and are the strongest candidates when deceptive features in the multiobjective space are expected. It is a common observation amongst most of the low dimensional objective functions (two or three variables) that using all the update techniques together is not necessarily the winning strategy. However combining at least one RSM and one non-RSM technique proves to work well. It is somewhat important to note that the Second NSGA2 layer shows its effect after sixth or seventh update iteration, as it needs time to converge and gather genetic information.

Update strategies that employ a greater variety of techniques prove to be more successful for functions with higher number of variables (25).

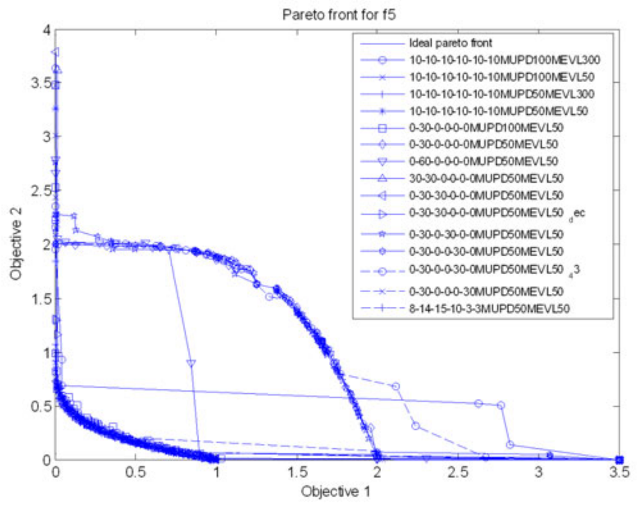

Fig. 7.5 Pareto front for F5

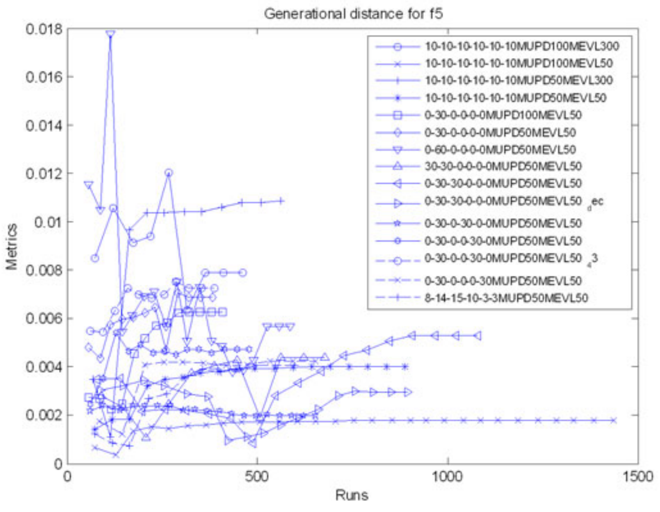

Fig. 7.6 Generational distance for F5

Fig. 7.9 and Fig. 7.10 show that the 'bump' function is particularly difficult for all strategies, which makes it a good test problem. This function has extremely tight constraint and multimodal features. It is not yet clear which of combination of strategies should be recommended, as the 'ideal' Pareto front has not been reached, however it seems that a decoupled secondary NSGA2 layer is showing a good

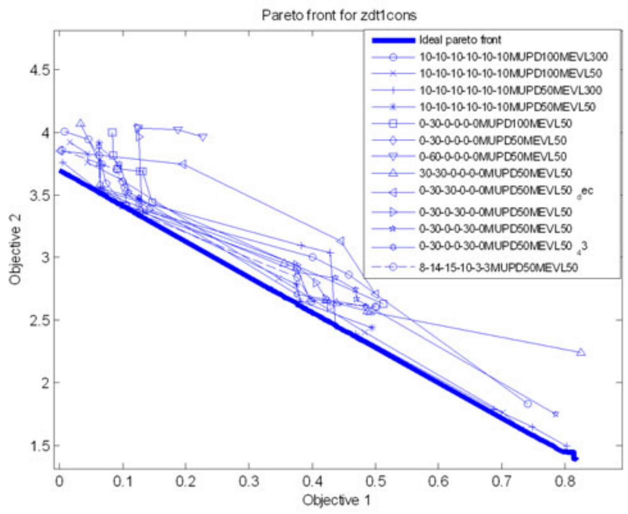

Fig. 7.7 Pareto front for ZDT1cons

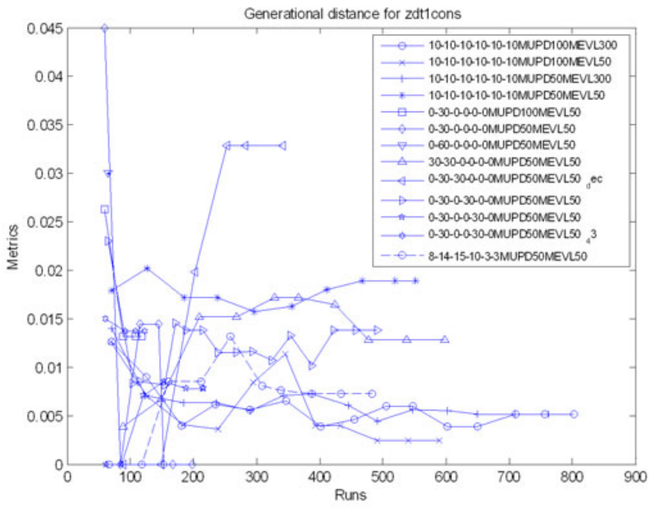

Fig. 7.8 Generational distance for ZDT1cons

advancement. We are continuing studies on this function and will give results in future publications.

To summarize the performance of each strategy an average statistics is computed. It is derived as follows. The actual performance in most cases is a trade-off between a given metric and the number of function evaluations needed for

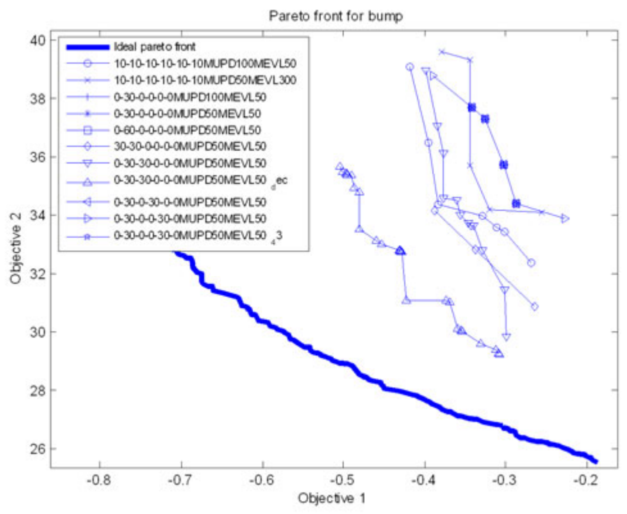

Fig. 7.9 Pareto front for the ‘bump’ function

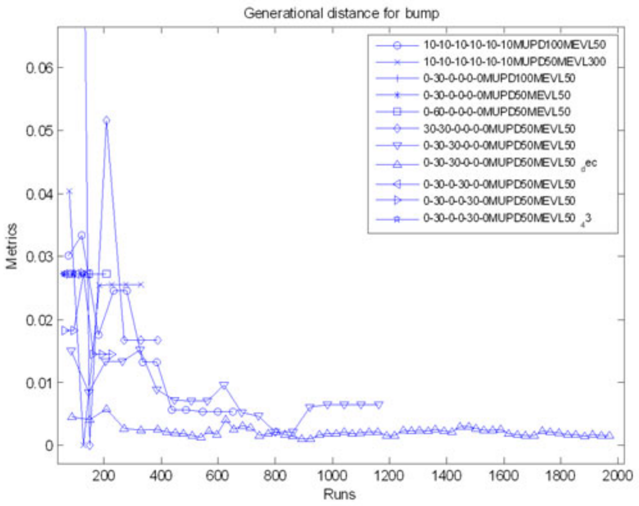

Fig. 7.10 Generational distance for the ‘bump’ function

convergence. Therefore the four metrics can be ranked against the number of runs, in the same way as ranks are obtained during NSGA2 operation. The obtained ranks are then averaged across all test functions. Low average rank means that the strategy has been optimal for more metrics and functions. These results are summarized in Table 14.2.

Table 7.2 Summary of performance

Random	*RSM PF*	*SL*	*RMSE*	*EI*	*KMEAN*	*Av. Rank*	*Min. Rank*	*Max. Rank*	*Note*
0	30	0	0	30	0	1.53	1	2	EI const.feas
0	30	30	0	0	0	1.83	1	3.33	SL coupled
0	30	0	30	0	0	2	1.33	3.33	RMSE
0	30	0	0	30	0	2.2	1.33	3	EI normal
0	30	30	0	0	0	2.8	1.33	4	SL decoupled
30	30	0	0	0	0	2.84	2	4	Random
0	60	0	0	0	0	2.85	2	3.33	RSM PF

The summary shows that all strategies are generally better than using only the conventional RSM based updates, which is expected, as the conventional method is almost always bound to converge at local solutions. However it must be underlined that a correct selection is problem dependant and must be selected with care and understanding.

7.9.4 *The Effect of the Initial Design of Experiments*

All methods presented here start from a given initial design of experiments. This is the starting point and this is what the initial surrogate model is based on. It is of course important to show the effect of these initial conditions. In what follows we have shown that effect by using a range of different initial DOEs. We have again

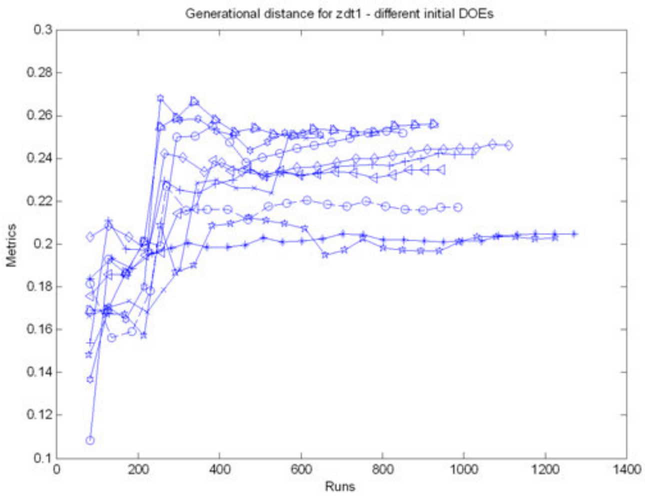

Fig. 7.11 Generational distance for zdt1 starting from different initial DOEs

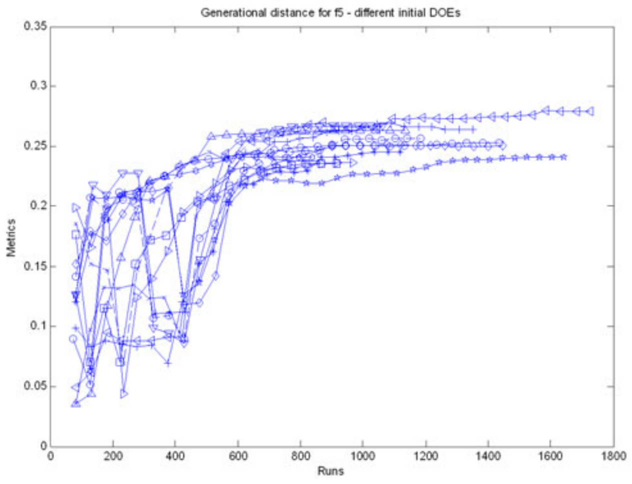

Fig. 7.12 Generational distance for F5 starting from different initial DOEs

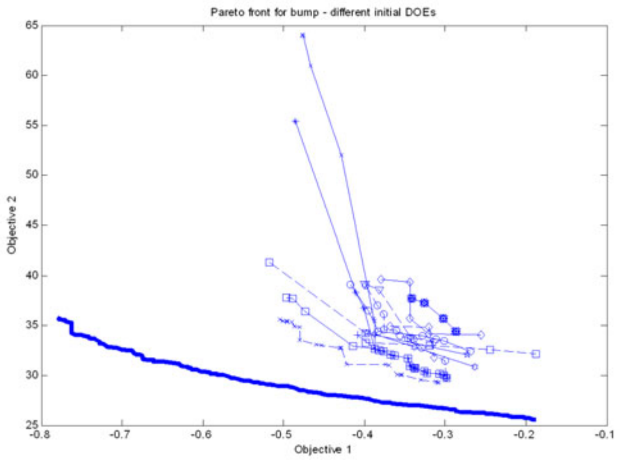

Fig. 7.13 Pareto fronts for 'bump' starting from different initial DOEs

used 10 updates for each of the techniques (60 updates per iteration in total) for all functions. The only difference being the starting set of designs.

Fig. 7.11 and Fig. 7.12 illustrate the generational distance for zdt1 and f5 functions - both with two variables. They both demonstrate a good averagibility,

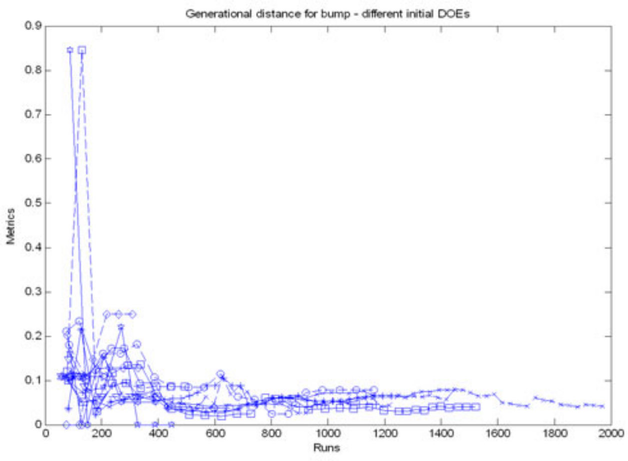

Fig. 7.14 Generational distance for 'bump' starting from different initial DOEs

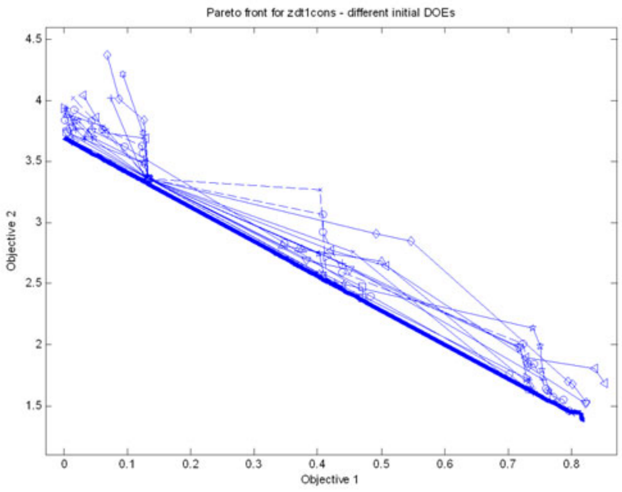

Fig. 7.15 Pareto fronts for 'zdt1cons' starting from different initial DOEs

confirming once again that the surrogate updates are fairly robust for functions with low number of variables.

Figures 7.13, 7.14 and 7.15 illustrate much greater variance and show that high dimensionality is a difficult challenge for surrogate strategies, however one should also consider the low number of function evaluations used here.

7.10 Summary

In this publication we have aimed to share our experience in tackling expensive multiobjective problems. We have shown that as soon as we decide to use surrogate models, to substitute for expensive objective functions, we need to consider a number of other specifics in order to produce a useful Pareto front. We have discussed the challenges that one might face when using surrogates and have proposed six update strategies that one might wish to use. Given understanding of these strategies, the researcher should decide on the budget of updates they could afford and then spread this budget over several update strategies. We have shown that it is best to use at least two different strategies – ideally a mixture of RSM and non-RSM based techniques. When solving problems with few variables we have shown that a combination of two or three techniques is sufficient, however with higher dimensional problems, one should consider using more techniques.

It is also beneficial to constrain the number of designs that are used for RSM training and also for RSM evaluation to limit the cost. The selection method of the designs then being used is open to further research. In this material we have used selection based on Pareto front ranking.

Our research also included parameters that reward the search for exploring the end points on the Pareto front. Although not explicitly mentioned in this material, our studies are using features such as improved crossover, mutation and selection strategies, declustering algorithm applied both in the variable and objective space to avoid data clustering. Data is also being automatically conditioned and filtered, and advanced kriging tuning techniques are used. These features are part of the OPTIONS [1], OptionsMATLAB and OptionsNSGA2_RSM suites [24].

Acknowledgements. This work was funded by Rolls – Royce Plc, whose support is gratefully acknowledged.

References

1. Keane, A.J.: OPTIONS manual, `http://www.soton.ac.uk/~ajk/options.ps`
2. Obayashi, S., Jeong, S., Chiba, K.: Multi-Objective Design Exploration for Aerodynamic Configurations, AIAA-2005-4666
3. Deb, K.: Multi-objective optimization using evolutionary algorithms. John Wiley & Sons, Ltd., New York (2003)
4. Zitzler, et al.: Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computational Journal 8(2), 125–148 (2000)
5. Knowles, J., Corne, D.: The Pareto archived evolution strategy: A new baseline algorithm for multiobjective optimisation. In: Proceedings of the 1999 Congress on Evolutionary Computation, pp. 98–105. IEEE Service Center, Piscatway (1999)
6. Fonseca, C.M., Fleming, P.J.: Multiobjective optimization and multiple constraint handling with evolutionary algorithms - Part II: Application example. IEEE Transactions on Systems, Man, and Cybernetics: Part A: Systems and Humans, 38–47 (1998)

7. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. Journal of Global Optimization 13, 455–492 (1998)
8. Sobol', I.M., Turchaninov, V.I., Levitan, Y.L., Shukhman, B.V.: Quasi-Random Sequence Generators, Keldysh Institute of Applied Mathematics, Russian Acamdey of Sciences, Moscow (1992)
9. Nowacki, H.: Modelling of Design Decisions for CAD. In: Goos, G., Hartmanis, J. (eds.) Computer Aided Design Modelling, Systems Engineering, CAD-Systems. LNCS, vol. 89. Springer, Heidelberg (1980)
10. Kumano, T., et al.: Multidisciplinary Design Optimization of Wing Shape for a Small Jet Aircraft Using Kriging Model. In: 44th AIAA Aerospace Sciences Meeting and Exhibit, Jannuary 2006, pp. 1–13 (2006)
11. Nain, P.K.S., Deb, K.: A multi-objective optimization procedure with successive approximate models. KanGAL Report No. 2005002 (March 2005)
12. Keane, A., Nair, P.: Computational Approaches for Aerospace Design: The Pursuit of Excellence (2005) ISBN: 0-470-85540-1
13. Leary, S., Bhaskar, A., Keane, A.J.: A derivative based surrogate model for approximating and optimizing the output of an expensive computer simulation. J. Global Optimization 30, 39–58 (2004)
14. Leary, S., Bhaskar, A., Keane, A.J.: A Constraint Mapping Approach to the Structural Optimization of an Expensive Model using Surrogates. Optimization and Engineering 2, 385–398 (2001)
15. Emmerich, M., Naujoks, B.: Metamodel-assisted multiobjective optimization strategies and their application in airfoil design. In: Parmee, I. (ed.) Proc of. Fifth Int'l. Conf. on Adaptive Design and Manufacture (ACDM), Bristol, UK, April 2004, pp. 249–260. Springer, Berlin (2004)
16. Giotis, A.P., Giannakoglou, K.C.: Single- and Multi-Objective Airfoil Design Using Genetic Algorithms and Artificial Intelligence. In: EUROGEN 1999, Evolutionary Algorithms in Engineering and Computer Science (May 1999)
17. Knowles, J., Hughes, E.J.: Multiobjective optimization on a budget of 250 evaluations. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 176–190. Springer, Heidelberg (2005)
18. Chafekar, D., et al.: Multi-objective GA optimization using reduced models. IEEE SMCC 35(2), 261–265 (2005)
19. Nain, P.: A computationally efficient multi-objective optimization procedure using successive function landscape models. Ph.D. dissertation, Department of Mechanical Engineering, Indian Institute of Technology (July 2005)
20. Voutchkov, I.I., Keane, A.J.: Multiobjective optimization using surrogates. In: Proc. 7th Int. Conf. Adaptive Computing in Design and Manufacture (ACDM 2006), Bristol, pp. 167–175 (2006) ISBN 0-9552885-0-9
21. Keane, A.J.: Bump: A Hard (?) Problem (1994), `http://www.soton.ac.uk/~ajk/bump.html`
22. Forrester, A., Sobester, A., Keane, A.: Engineering design via Surrogate Modelling. Wiley, Chichester (2008)
23. Yuret, D., Maza, M.: Dynamic hill climbing: Overcoming the limitations of optimization techniques. In: The Second Turkish Symposium on Artificial Intelligence and Neural Networks, pp. 208–212 (1993)
24. OptionsMatlab & OptionsNSGA2_RSM, `http://argos.e-science.soton.ac.uk/blogs/OptionsMatlab/`