NBI and MOGA-II, two complementary algorithms for Multi-Objective optimizations

Enrico Rigoni[1], Silvia Poles[2]

[1] ES.TEC.O S.r.l., Area Science Park,
Padriciano 99, 34012 Trieste, Italy
`enrico.rigoni@esteco.com`

[2] ES.TEC.O S.r.l., Area Science Park,
Padriciano 99, 34012 Trieste, Italy
`silvia.poles@esteco.com`

Abstract. The NBI-NLPQLP optimization method is tested on several multi-objective optimization problems. Its performance is compared to that of MOGA-II: since NBI-NLPQLP is based on the classical gradient-based NLPQLP, it is fast and accurate, but not as robust, in comparison with the genetic algorithm. Furthermore a discontinuous Pareto frontier can give rise to problems in the NBI's convergence. In order to overcome this problem, a hybridization technique coupled with a partitioning method is proposed.

Keywords. Genetic Algorithms, Normal-Boundary Intersection, Designs optimizations

1 Introduction

Multi-objective optimization algorithms, divided mainly into classical gradient-based approaches and stochastic methods, have been extensively studied and compared. The former group of methods is known to be fast and accurate but lacking in robustness, while the latter group is known to be very robust but requiring several steps to reach convergence.

In this paper NBI-NLPQLP and MOGA-II, which belong to the gradient-based methods and stochastic methods respectively, have been applied to the same test problem. The main purpose of this work was to study a hybrid technique coupled with a partitioning method; this technique combined the robustness of a genetic algorithm (where the discontinuity of the Pareto frontier gave rise to problems for NBI) with the speed of NBI to speed up the genetic algorithm.

2 Description of the algorithms

This section gives the reader a short description of the two algorithms employed. The work concentrates on these two algorithms since both are implemented in the commercial optimization software, modeFRONTIER [1]. Moreover, the multi-objective genetic algorithm (MOGA-II) has been demonstrated to perform as well as a well-tested evolutionary multi-objective methodology such as NSGA-II [2].

2.1 NBI-NLPQLP

The NBI-NLPQLP scheduler is a multi-objective scheduler based on the Normal-Boundary Intersection (NBI) method developed by I. Das and J. E. Dennis [3], coupled with the SQP algorithm developed by Prof. K. Schittkowski [4].

The NBI method applies to any generic smooth multi-objective problem, and it reduces the problem to many single-objective constrained subproblems, the so called *NBI subproblems*. The problem to be solved is at least subject to the restrictions imposed by the single-objective solver coupled with the NBI method, in this case the NLPQLP algorithm. For this reason the problem has to be at least smooth and well scaled.

Furthermore the NBI method imposes its own restrictions, since it needs a sufficiently regular Pareto curve in order to work properly.

This method starts considering each objective function separately, as an individual single-objective subproblem. Each subproblem is solved using the NLPQLP algorithm. Afterwards all the NBI subproblems are solved successively; the knowledge about the solution of the previous subproblem is used as the starting point for the next subproblem, in order to improve the algorithm convergence speed. In fact the NBI method orders efficiently the subproblems, in order that the solutions of two successive subproblems are expected to be close to each other. In this way the NBI subproblem is expected to converge in relatively few iterations. The number of NBI subproblems determines the "resolution" of the Pareto frontier. Clearly larger values for this parameter imply a better resolution of the Pareto frontier, at the cost of demand for more and more design evaluations.

Figure 1 shows graphically the working of the NBI method, and the aspect of one single NBI-subproblem. The problem has two objectives to be minimized, and the figure shows the objectives space.

The shaded area represents the region of feasible designs. The thick line is the Pareto frontier. The point A is the minimum of the first objective function f_1, while B minimizes the second objective f_2: the point F* is then the *utopia point*[1].

In this example the *convex hull of individual minima (CHIM)* corresponds to the line segment $\overline{\mathrm{AB}}$. Any NBI subproblem is specified giving its *barycentric coordinates (weights)*, β: for example the NBI subproblem outlined in the figure corresponds to $\beta = (0.75, 0.25)$. These values fix the position of the point H along the CHIM: in fact the components of the vector β are, respectively, the normalized lengths of the segments $\overline{\mathrm{BH}}$ and $\overline{\mathrm{AH}}$ (the normalization is made over the length of the segment $\overline{\mathrm{AB}}$).

[1] For the nomenclature used in this section refer to [3].

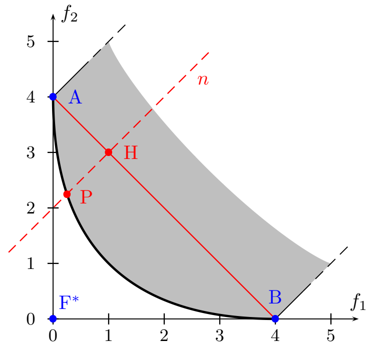

Fig. 1. The point P is the solution of the single-objective constrained NBI-subproblem outlined with the dashed red line n.

The line n is the *quasi-normal direction* passing through H, and it represents the constraints introduced by the NBI subproblem. The point P is then the solution of the single-objective constrained NBI-subproblem. The length of the segment $\overline{\mathrm{HP}}$ represents the new variable introduced by the NBI subproblem.

2.2 MOGA-II

MOGA-II is an improved version of MOGA (Multi-Objective Genetic Algorithm) by Poloni [5],[6],[7],[8] and is not to be confused with MOGA by Fonseca and Fleming [9] with which it shares only the same acronym. MOGA-II uses a smart multisearch elitism for robustness and directional crossover for fast convergence. Its efficiency is ruled by its operators (classical crossover, directional crossover, mutation and selection) and by the use of elitism.

Encoding in MOGA-II is done as in classical genetic algorithms and it uses four different operators for reproduction: classical crossover, directional crossover, mutation and selection. At each step of the reproduction process, one of the four operators is chosen (with regard to the predefined operator probabilities) and applied to the current individual.

Directional crossover assumes that a *direction of improvement* can be detected comparing the fitness values of two reference individuals. In [10] a novel operator called *evolutionary direction crossover* was introduced and it was shown that even in the case of a complex multimodal function this operator outperforms classical crossover.

The direction of improvement is evaluated by comparing the fitness of the individual Ind_i from generation t with the fitness of its parents belonging to generation $t-1$. The new individual is then created by moving in a randomly weighted direction that lies within the ones individuated by the given individual and his parents (see Figure 2). A similar concept can be however applied on the basis of directions not necessarily linked to the evolution but detected by selecting two other individuals Ind_j and Ind_k in the same generation.

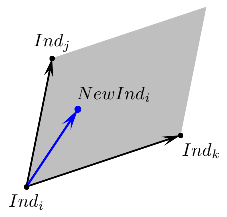

Fig. 2. Directional crossover between individuals Ind_i, Ind_j and Ind_k.

The selection of individuals Ind_j and Ind_k can be done using any available selection schema. In MOGA-II local tournament with random steps in a toroidal grid is used. First of all, the individual subject to reproduction is chosen as the starting point. Other individuals met in a random walk of assigned number of steps from that starting point are then marked as possible candidates for the first "parent" Ind_j. The list of all possible candidates for the second "parent" Ind_k is selected in the same way in a successive (and generally different) random walk from the same starting point. When the set of candidates is generated, the candidate with the best fitness is chosen. The number of steps N in the random walk remains fixed during the entire optimization run and is proportional to the population size.

The directional crossover operator has demonstrated to help the algorithm convergence for a wide range of numerical problems.

3 Benchmark problems

In this work the NBI-NLPQLP scheduler is tested on the following five benchmark problems.

The objective space plots of these problems are presented in Figure 3, on page 6: the feasible and unfeasible domains are clearly seen, as are the set of non-dominated solutions, i.e. the Pareto frontier.

3.1 no-hole

This example is the hole function problem [11] but *without the hole* [2], where the hardness parameter, h, is set to 2. The problem has two design variables and two objective functions to be minimized. The objectives are

$$\begin{aligned} f_1 &= (t+1)^2 + a \\ f_2 &= (t-1)^2 + a \ , \end{aligned} \tag{1}$$

[2] The hole is easily deleted by simply setting $b = 0$ (see Section 3.5, Eqs. 13).

where

$$t \in [-1, 1] \ , \quad a \in [0, 4] \ . \tag{2}$$

a and t are both function of the design variables x and y:

$$x \in [-1, 1] \ , \quad y \in [-1, 1] \ , \tag{3}$$

(see appendix A for details).

3.2 DEB

This constrained optimization problem is described in [12]. The problem is subject to two constraints, and it has two design variables and two objective functions to be minimized. The objectives are

$$\begin{aligned} f_1 &= x_1 \\ f_2 &= (1 + x_2)/x_1 \ , \end{aligned} \tag{4}$$

where

$$x_1 \in [0.1, 1] \ , \quad x_2 \in [0, 5] \ , \tag{5}$$

are the two design variables. The two linear constraints are

$$\begin{aligned} g_1 &= x_2 + 9x_1 \geq 6 \\ g_2 &= -x_2 + 9x_1 \geq 1 \ . \end{aligned} \tag{6}$$

3.3 TNK

This constrained test example, cited in [12], has two design variables, two objective functions to be minimized, and two non-linear constraints. The objectives are simply

$$\begin{aligned} f_1 &= x_1 \\ f_2 &= x_2 \ , \end{aligned} \tag{7}$$

where

$$x_1 \in [0, \pi] \ , \quad x_2 \in [0, \pi] \ , \tag{8}$$

while the constraints are

$$\begin{aligned} g_1 &= -{x_1}^2 - {x_2}^2 + 1 + 0.1 \cos(16 \arctan(x_2/x_1)) \leq 0 \\ g_2 &= (x_1 - 0.5)^2 + (x_2 - 0.5)^2 \leq 0.5 \ . \end{aligned} \tag{9}$$

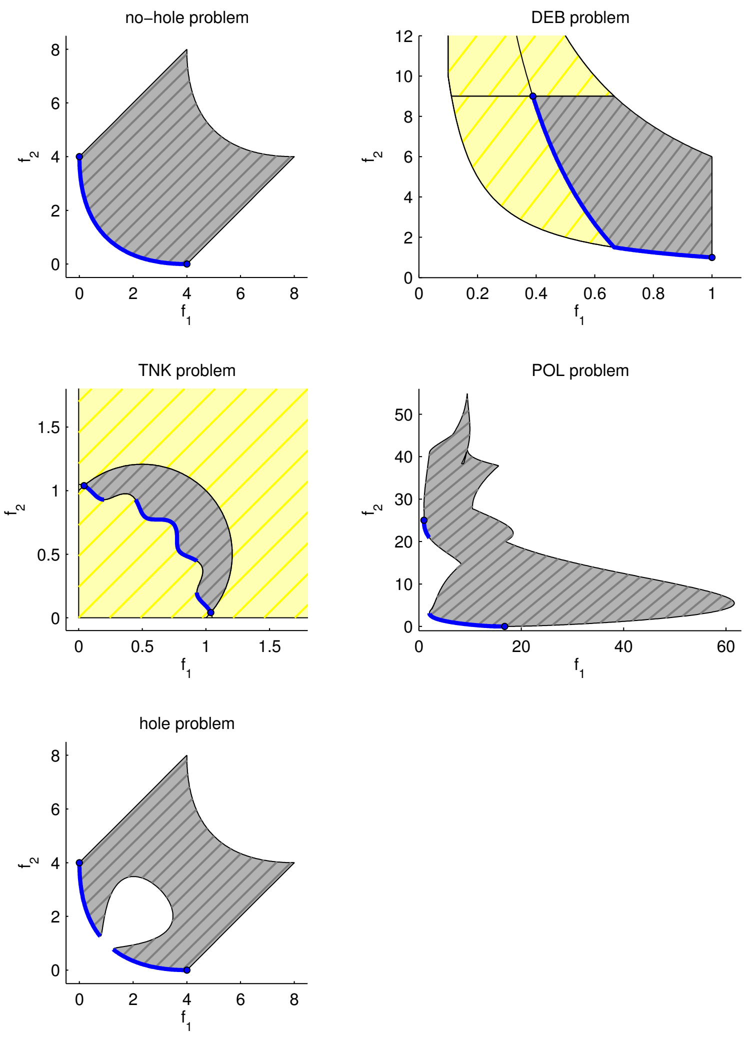

Fig. 3. The five benchmark problems' objective space. The gray shaded area represents the domain of feasible designs, while the yellow area depicts the unfeasible region of broken constraints. The dots are the minima for each single objective function; the thick blue curve represents the Pareto frontier, i.e. the sought solution.

3.4 POL

This is the Poloni test problem: in this case, the two objective functions to be minimized are

$$\begin{aligned} f_1 &= 1 + (a-b)^2 + (c-d)^2 \\ f_2 &= (x+3)^2 + (y+1)^2 \ . \end{aligned} \tag{10}$$

where the parameters present in f_1 are

$$\begin{aligned} a &= 0.5\sin(1) - 2.0\cos(1) + 1.0\sin(2) - 1.5\cos(2) \\ b &= 0.5\sin(x) - 2.0\cos(x) + 1.0\sin(y) - 1.5\cos(y) \\ c &= 1.5\sin(1) - 1.0\cos(1) + 2.0\sin(2) - 0.5\cos(2) \\ d &= 1.5\sin(x) - 1.0\cos(x) + 2.0\sin(y) - 0.5\cos(y) \ , \end{aligned} \tag{11}$$

and the variables ranges are

$$x \in [-\pi, \pi] \ , \quad y \in [-\pi, \pi] \ . \tag{12}$$

3.5 hole

This example is the hole function problem [11]: here the hole feature is fully implemented (compare with the no-hole example, Section 3.1, and see Figure 3). The hardness parameter, h is set to 2. The objectives are

$$\begin{aligned} f_1 &= (t+1)^2 + a + b\ \exp\left[-c\,(t-d)^2\right] \\ f_2 &= (t-1)^2 + a + b\ \exp\left[-c\,(t+d)^2\right] \ , \end{aligned} \tag{13}$$

where b, c, and d depend only on a (see appendix A for details).

4 Parameters setting

Unless otherwise stated, the parameters of the NBI-NLPQLP scheduler are set to these values:

```
Maximum Number of Iterations per Subproblem   : 500
Approximate Derivatives With                  : Forward Differences
Number of Pareto Points (Subproblems)         : 10
Final Termination Accuracy                    : 1.0E-5
Finite Difference Relative Perturbation       : 1.0E-7
Finite Difference Minimum Perturbation Policy : Constant
Constant Minimum Perturbation                 : 1.0E-7
```

Apart from the `Approximate Derivatives With` value, all the others are the default values.

The most important parameter for our purposes is the `Number of Pareto Points (Subproblems)`: the number of NBI subproblems determines the "resolution" of the Pareto frontier that the user wants to achieve. Larger values for this parameter imply a better resolution of the Pareto frontier, at the cost of demand for more and more design evaluations. In the benchmark problems we will change this parameter in order to analyse the results and the computational demand for different values.

Also the `Final Termination Accuracy` value has an important role: the default value, 10^{-5}, allows us to obtain good precision of results. The drawback is that such high accuracy could be prohibitive for some problems, inherently the problem's complexity and intrinsic precision: if this is the case, the user should relax that value, in order to allow NLPQLP to work properly. $N_{\mathrm{NBI}} = 10$

As with MOGA-II [2] — here used as a comparison — the design variables need to be discretized: a base value equal to 100001 is chosen for each variable. The algorithm's parameters are set to the default values, unless otherwise stated. The Design of Experiments (i.e. the set of initial designs, hereafter DOE) is the same as that used for NBI-NLPQLP, and the number of generations is chosen such that there will be almost the same number of evaluated designs, thereby guaranteeing a fair comparison.

5 Results

In this section the results are presented: for each problem there is a table, containing information on the problem settings and algorithm's performance. The legend below explains the symbols and abbreviations used in the tables.

- The column **param** specifies the parameter settings for each run.
  N_{NBI} : this is the `Number of Pareto Points (Subproblems)` parameter value (see Section 4).
  a : the `Final Termination Accuracy` parameter value (refer again to Section 4).
  T : this symbol indicates that there are some particular settings, which will be explained in detail below.
  P : this is the identity number of each single run, introduced for reference purposes.
- The column **DOE** specifies the Design Of Experiments scheme used in that specific run.
  n**R-**m (where n and m are two integers): n DOE designs are generated by means of the Random Sequence algorithm, with a `Random Generator Seed` equal to m.
  n**S** : n DOE designs generated by means of the pseudo random Sobol sequence.
  imp : stands for "improved" starting DOE. As described below, the scheduler requested a restart, suggesting the inclusion of some new designs in the original DOE. So these new entries are added in the DOE.
- The **result** column contains the scheduler exit status.
  OK : the scheduler ended regularly.
  F : the scheduler ended with `ERROR - FAILED INITIAL SEARCH`. The algorithm was not even able to find the starting minima for the single objective functions. Often in this case, the error is due to the NLPQLP solver, which exits with `ERROR - EXCEEDED THE MAX NUMBER OF ITERATIONS 20`. This case is symptomatic of a complex problem and/or a bad value for the `Final Termination Accuracy`: it is useful to try again decreasing the accuracy, i.e. increasing the parameter value (e.g., from a =1.0E-5 to a =1.0E-2 or higher).

I : the scheduler ended with `WARNING - Improved Starting Single-Objective Optima - Restart the scheduler including in the DOE the following designs ...` This means that the starting minima for the single objective functions were only local minima. Then it is useful to re-run the problem, including the suggested new entries in the DOE.

Bn (where n is an integer): the scheduler ended with `WARNING - FAILED` n `NBI Subproblems`. This means that n out of the total N_{NBI} subproblems failed.

***** : the algorithm managed to find the Pareto frontier only partially. In case of a discontinuous Pareto set, this means that some parts of the disconnected curve have not been mapped.

L : at least one of the starting minima for the single objective functions was only a local minimum. However, unlike case I, the scheduler was not able to recognize this.

– The column n_{eval} indicates the number of total evaluated designs (including in the count the evaluation of the initial DOE designs). This value quantifies the computational effort of the algorithm.
– The column n_{Pareto} shows the number of designs belonging to the Pareto set: it could be useful to estimate the success of the algorithm.

Note: it is worth putting a warning on the OK and Bn exit statuses. An OK result does not automatically guarantee that the algorithm achieved a good solution, in other words that it reached the complete Pareto set: in fact we must remember that NBI-NLPQLP is a local optimizer, and so can always get stuck in a local solution (see [13]). On the other hand, a Bn exit status — in the case that n is not a considerable fraction of N_{NBI} — should not be regarded necessarily as a bad result: even if some subproblems fail, others may be good solutions, even achieving a good mapping of the Pareto frontier.

5.1 no-hole

Even though this problem is well scaled, and the Pareto frontier is continuous and regular (see Figure 3), the problem is not so trivial. As stated in Section 3.1, the hardness parameter is set to 2: this choice makes the task harder (compare with [13]), "hiding" the Pareto set in the design space.

As shown in Table 1, with all parameters set to default values (P: 1–4), all the runs failed (F): this problem is a complex one, and it seriously engages the NBI-NLPQLP scheduler. Furthermore the default accuracy is probably well beyond the intrinsic precision of the problem. So the `Final Termination Accuracy` value has to be tuned, in order to make the algorithm work properly: a relaxed value for accuracy, say a =1.0E-2, is chosen (P: 5–8). With this choice we cannot avoid some Bn results — which show again the problem's intrinsic complexity — but at least we avoid complete failure.

P = 7 shows that a "bad" DOE (in this case the 10R-3 DOE), can confuse the algorithm, because one of the found starting minimum is a bad point. However, in this case the algorithm is able to recognize this event, and issues a warning that the problem should be rerun, including the suggested new entries in the DOE. The subsequent run, P = 7b, fully recovers the problem.

param	P	DOE	result	n_{eval}	n_{Pareto}
$N_{\text{NBI}} = 10$	1	10R-1	F	—	—
a =1.0E-5	2	10R-2	F	—	—
	3	10R-3	F	—	—
	4	10S	F	—	—
$N_{\text{NBI}} = 10$	5	10R-1	B2	1238	16
a =1.0E-2	6	10R-2	B1	680	21
	7	10R-3	I	—	—
	7b	“ imp	B1	763	16
	8	10S	OK	863	23
$N_{\text{NBI}} = 20$	9	10R-1	B3	2837	25
a =1.0E-2	10	10R-2	B6	2315	30
	11	10R-3	I	—	—
	11b	“ imp	B3	2317	26
	12	10S	B5	2743	27

Table 1. no-hole problem results.

In order to evaluate the computational demand of NBI-NLPQLP, we change the `Number of Pareto Points (Subproblems)` parameter, doubling its value, to 20. The results are presented in Table 1, along with the runs P: 9–12.

The comparison between the performance of NBI-NLPQLP and that of MOGA-II (for run P = 5) is shown in Figure 4a: NBI-NLPQLP maps the whole Pareto frontier while MOGA-II is able to map only a portion of the frontier.

In order to preserve consistency between results, MOGA-II was run using the same DOE (10R-1), while the number of generations was set such that the total (non-repeated) number of evaluated designs was as close as possible to that of the NBI-NLPQLP run.

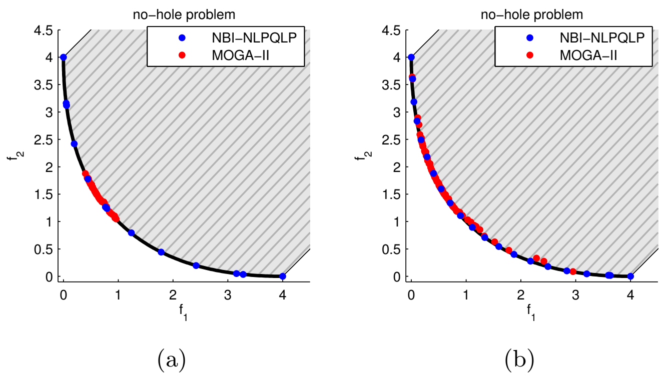

Fig. 4. (a) NBI-NLPQLP results (P = 5), vs. MOGA-II results. (b) NBI-NLPQLP results for the case P = 9, vs. MOGA-II results with 20R-1.

It is worth noting that in this case the performance of MOGA-II increases as we increase the number of DOE, using e.g. 20R-1, reducing consequently the number of generations (in order to keep constant the total number of evaluated designs). This is because this complex problem seems to demand a sufficiently large number of individuals, in order to avoid saturation of solutions. The spread of points of MOGA-II with 20R-1 is better than its run with 10R-1, even if the qualitative behaviour is the same: MOGA-II maps only a portion of the Pareto set.

Figure 4b shows the good spread of NBI-NLPQLP solutions for the run P = 9, i.e. for $N_{\text{NBI}} = 20$. MOGA-II was run with 20R-1, for the reason previously stated; MOGA-II densely covers the Pareto set, but only a portion of it.

5.2 DEB

In this problem the upper left limit of the Pareto frontier is given by one of the two constraints, while the border of the bottom part is caused by the domain definition (see Figure 3). This problem is not so simple because of constraints.

NBI-NLPQLP is very efficient in finding the true Pareto set, as shown in Table 2: with default parameters setting, all the runs (P: 1–4) ended regularly and successfully.

param	P	DOE	result	n_{eval}	n_{Pareto}
$N_{\text{NBI}} = 10$	1	10R-1	OK	95	55
	2	10R-2	OK	95	47
	3	10R-3	OK	92	51
	4	10S	OK	86	47
$N_{\text{NBI}} = 20$	5	10R-1	OK	137	84
	6	10R-2	OK	137	75
	7	10R-3	OK	134	79
	8	10S	OK	128	75

Table 2. DEB problem results.

Also with $N_{\text{NBI}} = 20$ (P: 5–9) the computational effort is limited, while the results are excellent.

The comparison between the performance of NBI-NLPQLP ($N_{\text{NBI}} = 10$, P = 1) and that of MOGA-II is presented in Figure 5a: since NBI-NLPQLP is very fast, it beats MOGA-II, given the same number of total evaluated designs. Obviously MOGA-II is also able to find the true Pareto frontier and to map it well, but it takes longer, since NBI-NLPQLP is a gradient based method.

This result is also more evident in Figure 5b, where it is shown the run P = 5 for NBI-NLPQLP, i.e. with $N_{\text{NBI}} = 20$.

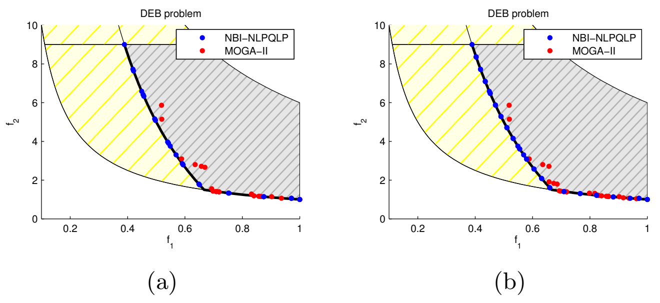

Fig. 5. (a) NBI-NLPQLP P = 1, vs. MOGA-II. (b) NBI-NLPQLP P = 5, vs. MOGA-II.

5.3 TNK

All the difficulties of this problem are due to the constraints. The shape of the margin between the feasible and unfeasible regions in the bottom left portion is particularly complex: it is such that it causes the Pareto frontier to be discontinuous (see Figure 3). This issue is of particular interest to our purposes: in fact we want to test the performance and robustness of NBI-NLPQLP in the presence of a non regular Pareto curve.

In spite of a discontinuous Pareto curve, NBI-NLPQLP performs well, as can be seen in Table 3. All runs ended regularly and successfully, both for $N_{\text{NBI}} = 10$ (P: 1–4), and for $N_{\text{NBI}} = 20$ (P: 5–9).

param	P	DOE	result	n_{eval}	n_{Pareto}
$N_{\text{NBI}} = 10$	1	10R-1	OK	132	9
	2	10R-2	OK	135	9
	3	10R-3	OK	135	9
	4	10S	OK	133	9
$N_{\text{NBI}} = 20$	5	10R-1	OK	217	20
	6	10R-2	OK	220	20
	7	10R-3	OK	220	20
	8	10S	OK	218	20

Table 3. TNK problem results.

This is better seen in Figure 6a: NBI-NLPQLP maps all three partial Pareto curves. The algorithm is very efficient in managing this task.

This is even more obvious in Figure 6b, with the run P = 5, i.e. $N_{\text{NBI}} = 20$. In this case the comparative difference between the performance of NBI-NLPQLP and that of MOGA-II is striking.

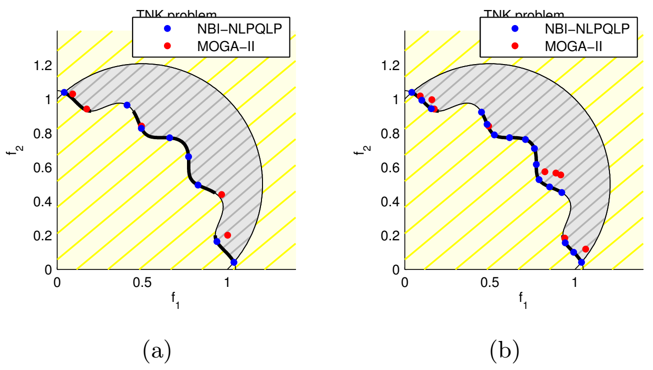

Fig. 6. (a) NBI-NLPQLP P = 1, vs. MOGA-II. (b) NBI-NLPQLP P = 5, vs. MOGA-II.

5.4 POL

This problem is a complex one, since the Pareto curve is discontinuous: the Pareto set is divided into two parts (see Figure 3). The situation here is quite different from the previous TNK case, where the discontinuity was caused by the shape of one constraint function. In fact here the discontinuity is generated intrinsically by the objective functions structure. So we can expect this problem to be harder.

As shown in Table 4, the NBI-NLPQLP results are not good: using default parameters setting all the runs (P: 1–4) present a variety of problems. It is worth analyzing them in detail.

param	P	DOE	result	n_{eval}	n_{Pareto}
$N_{\mathrm{NBI}} = 10$	1	10R-1	OK*L	108	79
	2	10R-2	B4*	231	83
	3	10R-3	B4*	233	97
	4	10S	B4*	249	97
$N_{\mathrm{NBI}} = 10$, T	5	10R-2	OK	197	55
$N_{\mathrm{NBI}} = 10$, T	6	—	OK	88	80
	7	—	OK	122	108
$N_{\mathrm{NBI}} = 20$, T	8	—	OK	145	134
	9	—	OK	197	183

Table 4. POL problem results.

The first run, P = 1, is different from others: in this case the scheduler ends with an apparent OK status. But reality is quite different: the problem is that the starting minimum for f_1 is only a local minimum (case L in Table 4). In fact afterwards NBI-NLPQLP is able to detect only the bottom part of the Pareto frontier (case marked * in Table 4). This can be seen in Figure 7-a, which shows run P = 1. This behaviour is due to an "unlucky" starting DOE (10R-1).

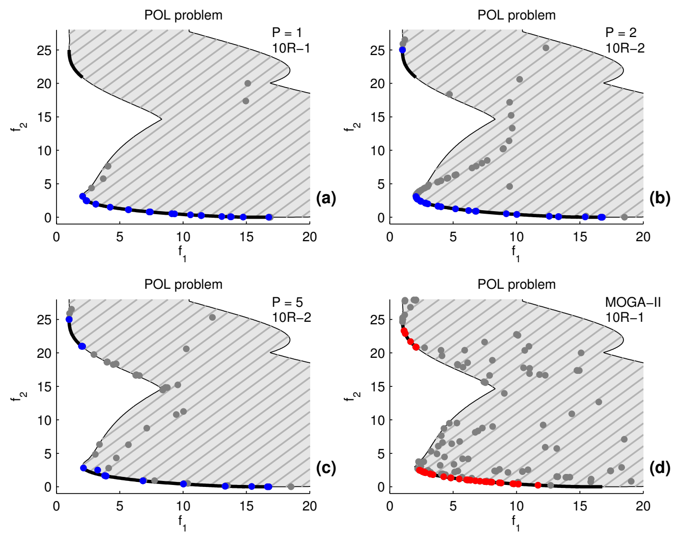

Fig. 7. NBI-NLPQLP for P = 1, 2, 5 cases, and MOGA-II with 10R-1 DOE and 23 generations (for a total of 199 non-repeated designs).

The failures in runs P: 2–4 are of a different nature: here NBI-NLPQLP finds the right starting global minima, but again fails to find the upper piece of the Pareto curve (* in table). The run P = 2 is shown in Figure 7-b. In Figure 7 all the evaluated designs are shown, plotted as gray dots, apart from the Pareto designs, plotted as coloured dots as usual. This fact highlights the problem: in solving the subsequent NBI subproblems — going from bottom right to top left — it is clear that the algorithm gets lost in the discontinuity. The algorithm is not able to jump the discontinuity, and so is unable to find the upper portion of the Pareto set. This fact has to be ascribed to the intrinsic complexity of the problem, in particular to the nature and shape of the discontinuity.

As a countercheck it is worth trying a test run: in P = 5, we use again 10R-2 as in P = 2, but instead of using the normal direct order in solving the subsequent series of NBI subproblems, we solve them in reverse order. The remarkable result shown in Table 4 (an OK run with both parts of the Pareto frontier mapped), can be directly seen in Figure 7-c: evidently going from top left to bottom right is a simpler task than solving the other way round. Clearly this feature is a characteristic of the POL problem. In general we cannot know a priori if it is better to go in one direction or another. So in the NBI-NLPQLP scheduler it is not up to the user to decide on the order of solution of the NBI subproblems. So we have to find a technique to overcome this problem, which will be presented below.

MOGA-II performance is shown in Figure 7-d: with 10R-1 DOE and 23 generations we have a total of 199 non-repeated evaluated designs, so this case is almost comparable with runs P = 2, and P = 5. MOGA-II results are by far the best ones, in terms of robustness. Consider also that 10R-1 was the DOE of run P = 1, where NBI-NLPQLP even failed to find the global minimum of f_1. So in this case the comparison between the performance of NBI-NLPQLP and that of MOGA-II works in favour of MOGA-II. MOGA-II is better because of its robustness, compared to the intrinsic fragility of NBI-NLPQLP — triggered in this POL problem by the nature of the Pareto curve discontinuity —.

Hybridization technique In this subsection we address the issue of combining the robustness of MOGA-II with the accuracy and speed of NBI-NLPQLP. This task is generated by the need to overcome the fragility of NBI-NLPQLP in the case of a discontinuous Pareto frontier: it is highly advisable to have a tool able to detect and map each single part of the Pareto curve well, without missing any portion.

In order to manage this problem, here we propose a **hybridization technique** coupled with a **partitioning method**.

The hybridization technique consists of combining a preliminary robust MOGA-II run with subsequent accurate NBI-NLPQLP runs. Since discontinuities can give rise to problems in NBI-NLPQLP convergence, we can isolate each single portion of the Pareto curve as an independent problem. We do this by introducing some new constraints in the objective space, extracting the relevant part of the Pareto set one at time: we call this procedure the partitioning method. We can do this by means of the knowledge of the problem we gained from the initial MOGA-II run. Each single isolated part of the Pareto curve can be now regarded as a good NBI-NLPQLP problem, since the local Pareto frontier is contiguous.

Let's apply this procedure to the POL problem, in order to see how it works. We can start from the MOGA-II run shown in Figure 7: from this robust run we can recognize the fact that the Pareto curve is separated into two distinct parts. We can treat them separately — one at time — by introducing, for example, the following constraints:

$$f_2 - 0.6 \cdot f_1 - 4 < 0 \ , \tag{14}$$

that allows the isolation of the bottom part, while

$$f_2 - 0.6 \cdot f_1 - 18 > 0 \ , \tag{15}$$

effectively separates the upper left part. These linear constraints are shown graphically in Figure 8 as dashed lines.

In Figure 8-a we simply report the results of MOGA-II, already presented in Figure 7, as red dots. The four green triangles are the designs we selected as

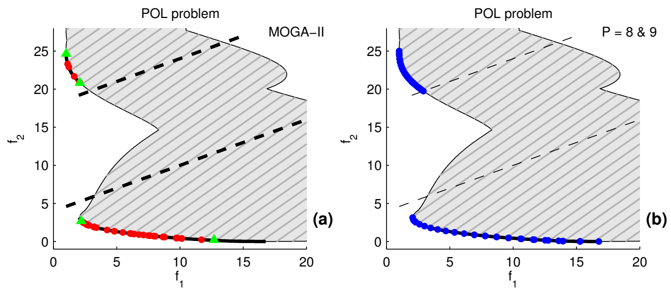

Fig. 8. Hybridization technique: (a) the results obtained with MOGA-II (red dots), and the points used as the starting DOE for the two subsequent separate NBI-NLPQLP runs (green triangles). The two dashed lines represent the newly introduced constraints. (b) the superimposition of the two NBI-NLPQLP runs P = 8 and P = 9.

DOE for the subsequent separate NBI-NLPQLP runs. The logic of this choice is clear: each couple of points spans the relevant portion of the Pareto curve to its maximal extension, as known so far.

The results are shown in Table 4: both for $N_{\mathrm{NBI}} = 10$ (P = 6, 7), and for $N_{\mathrm{NBI}} = 20$ (P = 8, 9) the mapping of each single Pareto portion ended regularly and successfully. Runs P = 6, 8 refer to the problem of finding the bottom part of the Pareto frontier (i.e. that one identified by the constraint equation 14), while runs P = 7, 9 isolate the upper left part of the Pareto set (Eq. 15). Note that for this latter case the n_{Pareto} column of Table 4 gives the number of designs belonging to the local Pareto set.

Results for the "high resolution" (i.e. $N_{\mathrm{NBI}} = 20$) cases P = 8, 9 are plotted in Figure 8-b: the quality of the NBI-NLPQLP mappings of the two portions is evident, in terms of both accuracy and uniformity of points distribution. It is worth noting that these subsequent NBI-NLPQLP runs are able to stretch the Pareto set portions to their full extent, even for the right part of the bottom portion, where the previous MOGA-II run was lacking.

5.5 hole

As seen in Section 3.5, the hole problem is only a complication of the no-hole problem of Section 3.1: the hole feature is now implemented, as shown in Figure 3; we can expect, therefore, even harder problems than those encountered in Section 5.1, faced now with the enhanced complexity caused by the introduction of a discontinuous Pareto frontier.

Keeping in mind the results of Section 5.1, we set the `Final Termination Accuracy` to a =1.0E-2 for all the runs presented in this section. This setting is implicitly assumed, and is not displayed in Table 5.

The results for all the runs with default parameter settings (P: 1–4) are not good (see table 5): NBI-NLPQLP is able to map only the bottom part of the Pareto frontier. It gets lost in the discontinuity, while solving the subsequent NBI subproblems, going from bottom right to top left.

param	P	DOE	result	n_{eval}	n_{Pareto}
$N_{\text{NBI}} = 10$	1	10R-1	B3*	756	32
	2	10R-2	B2*	680	18
	3	10R-3	B3*	632	17
	4	10S	B1*	708	9
$N_{\text{NBI}} = 10,$ T	5	10R-1	OK*	632	13
$N_{\text{NBI}} = 10,$ T	6	—	B2	665	31
	7	—	B1	491	59
$N_{\text{NBI}} = 20,$ T	8	—	B3	822	28
	9	—	B3	782	30

Table 5. hole problem results.

The same happens if we go the other way round: in P = 5, we again use 10R-1 as in P = 1, but we set the reverse order for NBI subproblems solving. Both P = 1 (direct order) and P = 5 (reverse order) are shown in Figure 9. Note that for P = 1 (Figure 9-a) the algorithm mapped only the bottom portion of the Pareto curve[3], while for P = 5 (Figure 9-b) it mapped only the top left portion. So now the situation is perfectly symmetrical: we cannot jump the unavoidable discontinuity, whichever direction we solve in. This is different from the POL problem, where the discontinuity was asymmetrical, and the direction we solved in made a difference (see Section 5.4, page 14).

As regards MOGA-II, results are shown in Figure 10. As seen in Section 5.1 for the no-hole problem, the performance of MOGA-II increases by increasing the number of DOE: so here we use 20R-1, instead of 10R-1. For this run, the number of generations has been set to 39, obtaining a total of 691 non-repeated evaluated designs: hence in this way results are directly comparable to those of NBI-NLPQLP (i.e. the runs P = 1 and 5 of Figure 9). The performance comparison clearly shows that MOGA-II results are better, since MOGA-II is able to map both portions of the Pareto frontier. It is worth noting that each single NBI-NLPQLP run is more accurate in finding its relevant Pareto part, with respect to MOGA-II, but fails in the overall task. MOGA-II is less accurate, but its robustness is the key feature of its success.

[3] Note that the points belonging to the upper part of the Pareto curve have been generated in the initial single-objective minimization of f_1, and not during a subproblem solving.

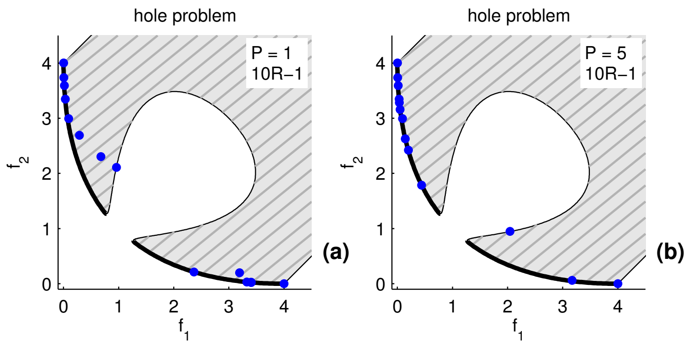

Fig. 9. (a) P = 1 run (direct order); (b) P = 5 (reverse order).

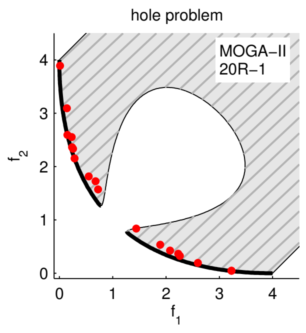

Fig. 10. MOGA-II run with 20R-1 DOE and 39 generations (for a total of 691 non-repeated designs).

Hybridization technique Given these premises, it is worth trying to combine the robustness of MOGA-II with the accuracy of NBI-NLPQLP, through the implementation of the hybridization technique.

For the initial MOGA-II run, instead of using the one shown in Figure 10, we can take an even "lighter" version, since we want to achieve only a good starting base for the subsequent NBI-NLPQLP runs. Again we use 20R-1 DOE, but with only 20 generations, for a total of 355 non-repeated evaluated designs. The results are shown in Figure 11-a.

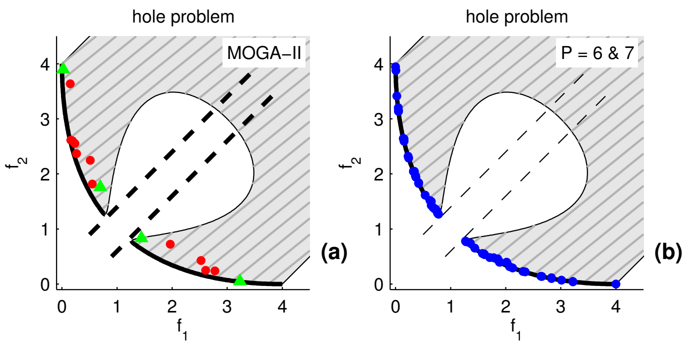

Fig. 11. Hybridization technique: (a) the results obtained with MOGA-II (red dots), and the starting DOE for the subsequent NBI-NLPQLP runs (green triangles). (b) the superimposition of the two NBI-NLPQLP runs P = 6 and P = 7. The two dashed lines represents the new introduced constraints.

The two independent Pareto frontier portions can be isolated introducing the following constraints:

$$f_2 - f_1 < -0.4 \ , \tag{16}$$

for the bottom part, and

$$f_2 - f_1 > 0.4 \ , \tag{17}$$

for the top left part. The constraints are presented in Figure 11 as dashed lines.

The DOE for the two subsequent NBI-NLPQLP runs are highlighted in green in the MOGA-II results, in Figure 11-a. The results for these two independent NBI-NLPQLP runs are presented in Table 5: runs P = 6, 7 refer to $N_{\text{NBI}} = 10$, while for runs P = 8, 9 the setting is $N_{\text{NBI}} = 20$. Runs P = 6, 8 relate to the bottom part problem, while runs P = 7, 9 relate to the top left part of the Pareto frontier.

On Figure 11-b the P = 6, 7 (i.e. $N_{\text{NBI}} = 10$) results are presented together: the two separated portions of the Pareto set are well represented. So also in this case the hybridization technique is successful.

Finally, a note of caution: the results for the cases P = 8, 9 (i.e. $N_{\mathrm{NBI}} = 20$) seem to deteriorate, instead of improving as expected. So this means that there is a residual intrinsic fragility in the procedure (pointed out also with other benchmarking conditions, not reported here), since results depend on problem conditions. This fragility cannot be totally eliminated. This fact should warn us that the hybridization technique can be regarded as a useful tool for improving the robustness of solutions, but cannot be considered as a panacea for all problem.

6 Conclusions

NBI-NLPQLP is an accurate and fast converging algorithm, since it is based on a classical gradient based method, i.e. NLPQLP. The drawback is its low robustness, especially when compared to that of MOGA-II.

The overall performance of NBI-NLPQLP on the benchmarking multi-objective optimization problems presented in this report is good: the algorithm is very efficient in finding the true Pareto frontier, and in mapping it almost uniformly. Even in one case of a discontinuous Pareto frontier (the TNK problem), NBI-NLPQLP is able to find and map uniformly all the portions of the Pareto curve. In other two cases (the POL and hole problems), characterized by a "strong" discontinuity in the Pareto frontier, we overcome the limited performance of NBI-NLPQLP by applying successfully an hybridization technique, combining the robustness of MOGA-II with the accuracy and speed of NBI-NLPQLP. This methodology consists of starting with a preliminary robust MOGA-II run, then isolating each single portion of the Pareto curve as an independent problem, each of which is treated with an independent accurate NBI-NLPQLP run.

While this paper was in preparation, Shukla and Deb [14] published a similar work comparing an evolutionary multi-objective optimizer with NBI and other classical methodologies. The conclusions of these authors differ in that they made a direct comparison between the performance of the two methodologies outlining the robustness of the evolutionary methods. Our results do not contradict these findings since we coupled the two methodologies within the hybridization framework in order to combine the robustness of the evolutionary methods with the speed and accuracy of NBI.

A Hole problem

In this appendix the hole functions problem details are presented, for sake of completeness. Refer to [11] for a full description.

The problem can be summarized as in the following scheme:

Hole problem
design variables
$x \in [-1, 1]$, $y \in [-1, 1]$
function parameters
$q = 0.2$, $p = 2$, $d_0 = 0.02$
problem hardness
$h = 2.0$
translation
$\delta = 1 - \sqrt{2}/2$
$x' = x + \delta$
$y' = y - \delta$
rotation of 45 deg
$\alpha = \pi/4$
$x'' = x' \cdot \cos\alpha + y' \cdot \sin\alpha$
$y'' = -x' \cdot \sin\alpha + y' \cdot \cos\alpha$
scale of π
$x''' = x'' \cdot \pi$
$y''' = y'' \cdot \pi$
change into problem coordinates
$u = \sin(x'''/2)$, $u \in [-1, 1]$
$v = \sin^2(y'''/2)$, $v \in [0, 1]$
apply hardness
$u' = u^h$ if $u \geq 0$; $u' = -(-u)^h$ if $u < 0$
$v' = v^{1/h}$
change into problem parameters
$t = u'$, $t \in [-1, 1]$
$a = v' \cdot 2p$, $a \in [0, 2p]$
other parameters computation
$b = (p - a)\exp(q)$ if $a \leq p$; $b = 0$ if $a > p$
$d = q/2 \cdot a + d_0$
$c = q/d^2$
objective functions
$\min f_1 = (t + 1)^2 + a + b\exp[-c\,(t - d)^2]$
$\min f_2 = (t - 1)^2 + a + b\exp[-c\,(t + d)^2]$

The problem has two design variables, x and y, and two objective functions to be minimized, f_1 and f_2. The parameter h controls the problem hardness: the higher h, the harder the problem.

The hole feature can easily be deleted by setting $b = 0, \quad \forall a$: this gives the no-hole problem.

References

1. modeFRONTIER version 3 Documentation. See also URL http://www.esteco.com

2. S. Poles. *Bench-marking MOGA-II.* Technical report 2004-001, ESTECO, Trieste, 2003, http://www.esteco.com.
3. Das, I., and Dennis, J. E. 1998, Normal-Boundary Intersection: A New Method for Generating the Pareto Surface in Nonlinear Multicriteria Optimization Problems, SIAM Journal on Optimization, 8(3), 631
4. Schittkowski, K. 2001, NLPQLP: A New Fortran Implementation of a Sequential Quadratic Programming Algorithm - User's Guide, Report, Department of Mathematics, University of Bayreuth
5. G. Mosetti and C. Poloni. Aerodynamic shape optimization by means of a genetic algorithm. *5th International Symposium on Computational Fluid Dynamics*, Sendai, Japan, 1993.
6. C. Poloni and V. Pediroda. GA coupled with computationally expensive simulations: tools to improve efficiency. In *Genetic Algorithms and Evolution Strategies in Engineering and Computer Science*, pages 267–288, John Wiley and Sons, England, 1997.
7. C. Poloni, A. Giurgevich, L. Onesti, and V. Pediroda. Hybridization of a multiobjective genetic algorithm, a neural network and a classical optimizer for a complex design problem in fluid dynamics. *Computer Methods in Applied Mechanics and Engineering*, volume 186, pages 403–420, 2000.
8. D. Spicer, J. Cook, C. Poloni and P. Sen. EP20082 Frontier: Industrial Multi-Objective Design Optimisation. In *Proceedings of the 4th European Computational Fluid Dynamics Conference (ECCOMAS 98)*, John Wiley and Sons, England, 1998.
9. C. M. Fonseca and P. J. Fleming. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization.
10. K. Yamamoto and O. Inoue. New evolutionary direction operator for genetic algorithms. *AIAA Journal*, volume 33, number 10, pages 1990–1993, 1995.
11. Rigoni, E. 2004, Hole functions problem, ESTECO Technical Report 2004-002, http://www.esteco.com.
12. Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. 2000, A Fast and Elitist Multi-Objective Genetic Algorithm: NSGA-II, KanGAL Report Number 2000001 In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423, San Mateo, California, 1993.
13. Rigoni, E. 2004, NBI-NLPQLP Scheduler, ESTECO Technical Report 2004-003, http://www.esteco.com.
14. P. Shukla, K. Deb, S. Tiwari (2005). Comparing classical generating methods with an evolutionary multi-objective optimization method. Proceedings of the Third International Conference on Evolutionary Multi-Criterion Optimization (EMO-2005). Guanajuato, Mexico. Lecture Notes on Computer Science 3410, pages 311-325.