Evolutionary Multi-Objective Optimization Without Additional Parameters

Kalyanmoy Deb
Department of Mechanical Engineering
Indian Institute of Technology Kanpur
Kanpur, PIN 208016, India
Email: deb@iitk.ac.in
http://www.iitk.ac.in/kangal/deb.htm

KanGAL Report Number 2006003

Abstract

The present-day evolutionary multi-objective optimization (EMO) algorithms had a demonstrated history of evolution over the years. The initial EMO methodologies involved additional niching parameters which made them somewhat subjective to the user. Fortunately, soon enough parameter-less EMO methodologies have been suggested thereby making the earlier EMO algorithms unpopular and obsolete. In this paper, we present a functional decomposition of a viable EMO methodology and discuss the critical components which require special attention for making the complete algorithm free from any additional parameter. A critical evaluation of existing EMO methodologies suggest that the elitist non-dominated sorting GA (NSGA-II) is one of EMO algorithms which does not require any additional implicit or explicit parameters other than the standard EA parameters, such as population size, operator probabilities, etc. This parameter-less property of NSGA-II is probably the reason for its popularity to most EMO studies thus far.

1 Introduction

Evolutionary algorithms (EAs) came out as serious contenders in optimization studies during the past two decades. EAs have certain features (such as, flexible operators, no need for using gradients, ease in tackling mixed-integer problems and combinatorial problems, etc.) which are rare to be found in any other single optimization method, including classical methods. However, the flexibilities in their usage come with some onus on the part of the user. To put different flexible EA operators together to make an effective search, they have to be tuned properly as to make a proper balance [22, 19] of two conflicting aspects needed in a successful optimization algorithm: (i) exploitation of available resource and (ii) exploration of search space. The former aspect deals with the selection operator and is related to the selection probabilities assigned to better population members for them to be used as parents. The more the assigned selection probability, the larger is the exploitation of available resource. But since a population early on need not have the optimal solution, too much exploitation may lead to a premature convergence of the overall algorithm. Whether a premature convergence actually takes place or not is also closely related to the associated explorative search power of the algorithm. This aspect is related to the variation (recombination and mutation) operators used in an EA. If these operators are designed in a way to have a large search power (large ergodicity, as defined by Radcliffe [32]), a large exploitation may be allowed to constitute a successful search [22]. Thus, it is clear that a successful application of an EA is far from being simple and a proper balancing act of exploitation-exploration issue is

essential in an EA. The balance between these issues is controlled by a proper choice of different EA parameters (such as, population size, selection pressure parameter, EA operators and their probabilities, etc.). Unfortunately, different optimization problems require different values of some or all of these parameters and a successful application of an EA often requires a proper tuning of these parameters from problem to problem. Since these parameters are essential to be fixed in any form of an EA, it is not surprising that these parameters must have to be set properly in a successful application of a multi-objective EA.

As the name suggests, multi-objective optimization deals with multiple conflicting objectives and usually the optimal solution of one of the objectives is not necessarily the optimum for any of the other objectives. In such a scenario, instead of one optimal solution, a number of solutions are optimal. These solutions are called Pareto-optimal solutions. In the growing literature on evolutionary multi-objective optimization (EMO) [5], the task is to first find a well-distributed set of Pareto-optimal solutions and then, based on the trade-off information about the solutions, choose one specific solution for implementation [10].

Intuitively, EMO algorithms have a more stringent task to be achieve than the single-objective EAs. While in most single-objective EAs, the task is to find a single optimal solution, an EMO is expected to find a set of Pareto-optimal solutions causing an optimal trade-off among multiple conflicting objectives. Besides the requirement of these optimized solutions to be as close to the true Pareto-optimal front as possible, they also have an additional requirement of being well distributed over the entire Pareto-optimal region. The task of finding multiple, well spread-out optimal solutions resembles the task in a niching EA, in which the target is to find multiple optimal solutions in a multi-modal, single-objective optimization problem [23, 11]. Besides, an EMO algorithm must have another important consideration – the need of assigning an appropriate implicit or explicit fitness measure so that solutions can progress towards the true Pareto-optimal frontier. All these operations seem demanding and it is not obvious whether an EMO algorithm can be devised without the need of any additional parameter.

In this paper, we first discuss the nature of optimal solutions in a multi-objective optimization problem and then outline some standard EA parameters which are usually associated with any single-objective EA. Thereafter, we attempt to decompose different tasks necessary to a multi-objective EA functionally into three major components and discuss the difficulties which one can face in trying to devise them as parameter-free procedures. Finally, we consider a number of existing EMO methodologies and show that a specific algorithm has a systematic implementation which allowed the complete algorithm to have no additional parameter. This specific algorithm (the elitist non-dominated sorting GA or NSGA-II [7, 8]) is by far the most popular EMO methodology used and cited in the EMO literature. Through a comparison of NSGA-II with other contemporary EMO algorithms according to their requirement of additional parameters, we argue that the parameter-less approach of NSGA-II is one of the main reasons for its success and popularity among researchers and practitioners. The systematic discussion for achieving parameter-less algorithms presented in this paper should encourage readers to develop more effective parameter-less EMO or EA approaches.

2 Multi-Objective Optimization

A multi-objective optimization problem involves a number of objective functions which are to be either minimized or maximized. As in the single-objective optimization problem, the multi-objective optimization problem usually has a number of constraints which any feasible solution (including the optimal solution) must satisfy. In the following, we state the multi-objective

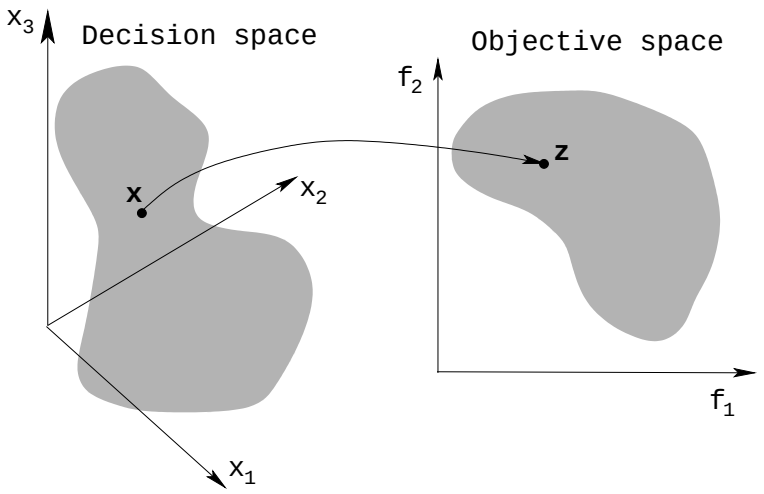

Figure 1: Representation of the decision variable space and the corresponding objective space.

optimization problem (MOOP) in its general form:

$$\left.\begin{array}{rll} \text{Minimize/Maximize} & f_m(\mathbf{x}), & m = 1, 2, \ldots, M; \\ \text{subject to} & g_j(\mathbf{x}) \geq 0, & j = 1, 2, \ldots, J; \\ & h_k(\mathbf{x}) = 0, & k = 1, 2, \ldots, K; \\ & x_i^{(L)} \leq x_i \leq x_i^{(U)}, & i = 1, 2, \ldots, n. \end{array}\right\} \tag{1}$$

A solution $\mathbf{x}$ is a vector of n decision variables: $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$. The solutions satisfying the constraints and variable bounds constitute a *feasible decision variable space* $\mathcal{S}$, or simply the variable space. One of the striking differences between single-objective and multi-objective optimization is that in multi-objective optimization the objective functions constitute a multi-dimensional space, in addition to the usual variable space. For each solution $\mathbf{x}$ in the variable space, there exists a point in the objective space, denoted by $\mathbf{f}(\mathbf{x}) = \mathbf{z} = (z_1, z_2, \ldots, z_M)^T$. A mapping exists between an n-dimensional solution vector and an M-dimensional objective vector through the objective function, constraints, and variable bounds. Figure 1 illustrates these two spaces and a mapping between them.

Although the fundamental difference between single and multiple objective optimization lies in the cardinality in the optimal set, from a practical standpoint a user needs only one solution, no matter whether the associated optimization problem is single-objective or multi-objective. In the case of multi-objective optimization, the user is now in a dilemma. Which of these optimal solutions must one choose? This is not an easy question to answer. It involves many higher-level information which are often non-technical, qualitative and experience-driven. However, if a set of many trade-off solutions are already worked out or available, one can evaluate the pros and cons of each of these solutions based on all such non-technical and qualitative, yet still important, considerations and compare them to make a choice. Thus, in a multi-objective optimization, ideally the effort must be made in finding the set of trade-off optimal solutions by considering all objectives to be important. After a set of such trade-off solutions are found, a user can then use higher-level qualitative considerations to make a choice. In view of these discussions, the following principle was suggested as an *ideal multi-objective optimization procedure*:

Step 1 Find multiple trade-off optimal solutions with a wide range of values for objectives.

Step 2 Choose one of the obtained solutions using higher-level information.

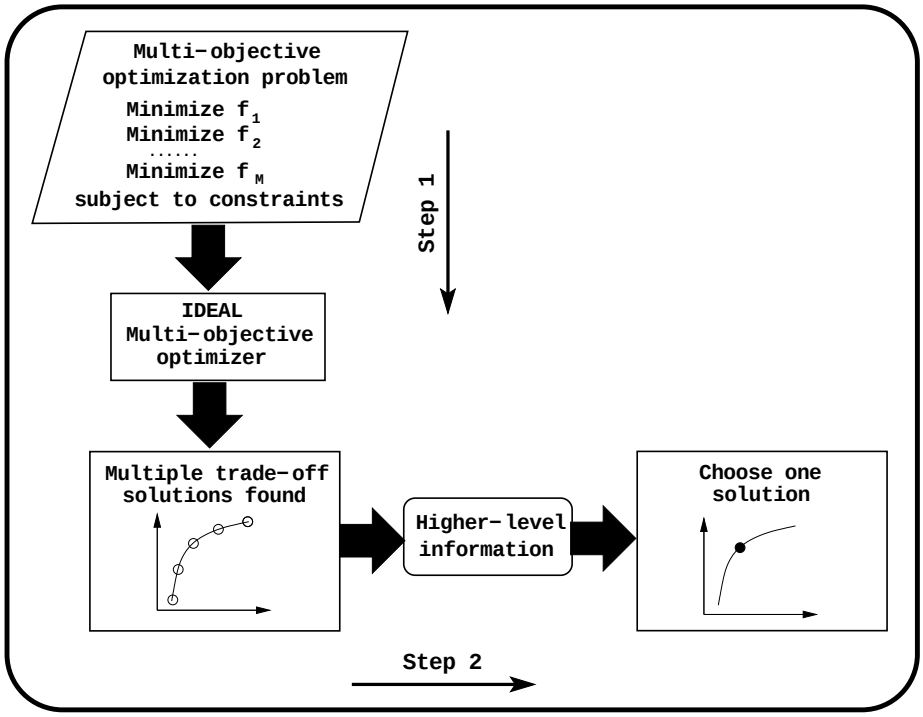

Figure 2: Schematic of an ideal multi-objective optimization procedure.

Figure 2 shows schematically the principles in an ideal multi-objective optimization procedure. In Step 1 (vertically downwards), multiple trade-off solutions are found. Thereafter, in Step 2 (horizontally, towards the right), higher-level information is used to choose one of the trade-off solutions. With this procedure in mind, it is easy to realize that single-objective optimization is a degenerate case of multi-objective optimization. In the case of single-objective optimization with only one global optimal solution, Step 1 will find only one solution, thereby not requiring us to proceed to Step 2. In the case of single-objective optimization with multiple global optima, both steps are necessary to first find all or many of the global optima and then to choose one from them by using the higher-level information about the problem. Realizing this degeneracy between single and multi-objective optimization, we have designed an *omni-optimizer*, which is capable of solving different types of optimization problems [14]. We shall discuss this generic procedure in Section 7, but first enumerate the necessary components of an evolutionary multi-objective optimization.

2.1 Evolutionary Principles

Since an evolutionary algorithm deals with a number of population members in each generation, an EA is an ideal candidate for finding multiple Pareto-optimal solutions in a multi-objective optimization problem. In Sections 4 to 7, we shall discuss a few modified EAs for doing this task, but all of these methods perform the following tasks:

1. Emphasize *non-dominated* solutions for progressing towards the Pareto-optimal front.
2. Emphasize *less-crowded* solutions for maintaining a good diversity among obtained solutions.
3. Emphasize *elites* to provide a faster and reliable convergence near the Pareto-optimal front.

Different algorithms implement the above three features differently, but they are mostly implemented in the *selection* operator of an EA. They can be used while choosing parent solutions for recombination and they can also be used while deciding to accept an offspring solution into the population. Potentially, a naive implementation may involve setting additional parameters for each of the above features, but intuitively the latter two tasks (determining less-crowded solutions and determining elite population members) may become difficult to implement without any user-defined parameter. We discuss these two matters in the following paragraphs.

The less-crowded solutions are those which are not surrounded by too many solutions. First of all, the neighborhood can be defined in either of the two spaces: (i) objective space and (ii) variable space. Secondly, how many neighboring solutions are too many? The answer to this question may require a parameter to resolve. The elite solutions are those which are best in the population. In a multi-objective problem, how does a solution define to be the best, when there are multiple conflicting objectives? This question can be answered somewhat by declaring all non-dominated solutions to be the best. As shown elsewhere [5], problems having a large number of objectives, almost all population members belong to the non-dominated front. It is also natural to understand that not all solutions can be declared as elites and be passed on to the next population. This leaves us with a question of which or what proportion of non-dominated solutions are to be redefined as elites? This involves setting another parameter.

Thus, it is easy to think of an evolutionary multi-objective optimization algorithm which will have components to take care of the above three essential tasks, but such an algorithm may involve a number of additional parameters which the user must have to set. It becomes a challenging task to come up with an EMO algorithm which would perform all three tasks efficiently but without any additional parameter. We shall discuss how parameter-less implementation of such salient tasks is achieved in a few popular EMO methodologies later in this study. But before we do this, we would like to mention a few essential parameters which a single-objective EA run usually demands. These parameters are also required to be supplied in running most multi-objective EA algorithms.

3 Essential Parameters in an Evolutionary Algorithm

Many chapters of this book have dealt with the essential parameters needed in a single-objective evolutionary algorithm (EA) and suggested various procedures of setting up the adequate values of these parameters. Here, we briefly mention the essential parameters often needed to be fixed in an EA:

1. Population size (N),
2. Number of generations (T),
3. Parameters related to Selection (selection pressure (s), selection operator, etc.),
4. Parameters related to recombination (crossover probability (p_c), crossover operator, etc.),
5. Parameters related to mutation (mutation probability (p_m), mutation operator, etc.),
6. Generation gap parameter (G),
7. Number of independent runs of an EA.

Population size is a crucial parameter in the successful application of an EA. For an application without any limit in the number of overall function evaluations, there exists a *critical* population size below which the EA is not expected to perform to its best and on or beyond of which the EA

performs at its best. However, additional population members demand more function evaluations compared to that needed by the EA which uses the critical population size. However, if an EA is applied for a limited number of function evaluations, there exists a critical range of population sizes below and above of which the performance of an EA gets degraded. Thus, in such cases, it is essential to choose a population size as close to the critical population size as possible. Some statistics-based estimation of the size of a random initial population exist [21, 24, 33] and some numerical procedures do exist as well [1, 35].

Even with an adequate population size and other EA operators, the EA must be run for a *critical* number of generations for a convergence near the optimal solution. For some problems (mostly test problems), a bound on number of generations is computed from a complexity analysis [40]. However, such an analysis is usually problem-specific and cannot be generalized for any arbitrary problem [41].

An EA is a flexible optimization algorithm, because it allows the user to choose any suitable EA operators. Since it is flexible, it also puts an onus on the part of the user to choose *proper* operators. There are mainly two different EA operators – a selection operator mimicking the Darwin's survival of the fittest principle and one or more variation operators causing the generation of a new offspring population. Usually, two variation operators are used in most applications: (i) recombination or crossover operator and (ii) mutation operator. Each of these operators have many variations and each variation may involve different additional user-defined parameters. For example, the following is a list of selection operators popularly used:

1. s-ary tournament selection, in which s solutions are picked with or without replacement and the best solution (in terms of their fitness values) is selected.

2. Ranking selection, in which population members are ranked according to the fitness values and a higher rank (better solution) solution is selected proportionately more often (by assigning s copies to the best solution).

In these selection operators, the parameter s is loosely called the selection pressure and the performance of an EA procedure depends on the choice of this parameter s. Goldberg and Deb [20] computed the selection pressure for the above two operators and a few others.

Similarly, different recombination operators and their associated parameters affect the performance of an EA and must be chosen properly. One common parameter among different crossover operators is the 'probability of crossover' (p_c), which denotes the expected proportion of the population which participates in the crossover operation. However, more operator-specific parameters exist and must also be set by the user. For example, the binary-coded crossover operator requires the user to specify the number of cross-sites. For real-parameter recombination operators such as SBX [6], BLX [16] and FR [39] require a parameter, controlling the extent of the search, whereas some recombination operators such as PCX [9], UNDX [29], SPX [25], differential evolution (DE) [38], operator in particle swarm optimization (PSO) [28] require more than one parameters to be set by the user.

Mutation operator also involves a probability of mutation p_m to be set. In addition, at least one mutation-specific parameter is often required to be supplied by the user.

Besides, the generation gap is an important matter which controls the overall selection pressure per function evaluation. On one extreme (called the 'generational EA'), the generation gap can be equal to the population size, meaning that all N offspring solutions must be created and evaluated before any of them is put back into the population. On the other extreme (called the 'steady-state EA'), the generation gap is one, meaning that after every new offspring solution is created, it is considered for its inclusion to the population. The latter causes a larger selection pressure and the EA is often termed as an 'greedy EA'. Other intermediate propositions are certainly possible [34, 27].

3.1 Desired and Undesired Parameters

In most optimization studies, an algorithm involving parameters which the users are required to be set is considered negatively, simply because of the obtained optimized solution may be sensitive to the parameter values. In fact, more the number of parameters involved in the algorithm, the worse is the situation. Often, good optimization studies make a parametric analysis and present the sensitivities of each parameter on the obtained result. Often researchers suggest ways to estimate parameters which are found to have worked well on known test problems.

However, there may exist certain parameters which are *desired* in an algorithm and provide the user (a designer or an applicationist) flexibility and control in the obtained solution. Although the outcome of the optimization depends on these tunable parameters, instead they being controlled by the developers of the algorithm, the users prefer to choose them on their own. In multi-objective optimization algorithms, such tunable parameters (for example, ϵ in the ϵ-MOEA [12] or a fixation of one or more reference points in a reference-point based EMO [13]) allows a decision-maker to find solutions closer to the desired region on the Pareto-optimal frontier. We call these parameters as 'desired parameters' for a problem solving task.

4 NSGA, Contemporary Algorithms, and Additional Parameters

In this section, we investigate the non-elitist EMO methodologies for their requirement of any additional parameter for achieving two tasks: (i) in determining non-dominated solutions and (ii) in determining less-crowded solutions. Since these methods did not use any elite-preservation operator, we ignore the third task in these first-generation EMO algorithms. A detail description of these algorithms can be found in [5].

Non-dominated sorting GA or NSGA [37] and multi-objective GA or MOGA [18] emphasized non-dominated solutions without any additional parameter, but the niched-Pareto GA or NPGA [26] introduced a subpopulation size parameter for defining the non-dominated solutions. This parameter is required to be set by the user.

Next, we discuss their requirement for additional parameter in implementing the niching operator. The non-elitist EMO methodologies, which were able to find a set well-distributed near-Pareto-optimal solutions, all used an explicit niche-preserving operator involving a niching parameter. Thus, these methodologies demanded an additional parameter on top of EA parameters mentioned above. For example, the non-dominated sorting GA or NSGA [37] required the niching parameter σ_{share}, which defined the maximum Euclidean distance among two solutions in the variable space for them to qualify as solutions from the same niche. Although a subsequent study [4] has suggested a procedure of estimating this parameter, the performance of NSGA was shown to depend on the choice of this parameter [37]. The purpose of the niching parameter is to define the maximum distance between any two solutions to remain in a single niche. If a small niching parameter is used, fewer solutions will qualify to remain in a single niche, thereby clustering all solutions to more number of niches. The effect of choosing a small niching parameter in an EMO is, therefore, to converge to many Pareto-optimal solutions. There are some difficulties in using a fixed niching parameter throughout the run. In most problems, the search region where an algorithm is started is often wider than the Pareto-optimal region in which the algorithm is expected to converge at the end. By keeping the same niching parameter throughout, many different niches must have to be encountered early on to keep a good distribution of solutions, thereby requiring a larger population size. However, near the Pareto-optimal region, a large population may not be required and with a large population, multiple solutions will exist in each niche. Although the use of population-best and population-worst function values can be used to normalize the objective values and an identical niching parameter can be used throughout, this idea causes another difficulty. Early on, the effective niche size will be large for a similar argument given above and

there may not be enough representative solutions to search the region effectively. Thus, for both of the above cases (with or without normalization), there is a need of a variable population size to make an effective search. Although an adaptive change of population size with generation is in order, most EMO methodologies use a fixed population size.

Similarly, MOGA [18] required an identical niching parameter as in NSGA, but the distance measure was computed in the objective space. The study suggested a clever procedure for estimating the niching parameter and a later study [5] suggested a normalized version of the procedure and presented a table showing the σ_{share} value as a function of a fixed population size and number of objectives. However, these estimation procedures are not free from approximations and some tuning may be necessary for their proper working in an arbitrary problem.

NPGA [26] required two parameters: σ_{share} and an additional parameter t_{dom} – the number of population members chosen for counting the number of similar solutions to two competing solutions. The original study did not provide any specific guidelines for estimating these parameters.

The need for these extra parameters for multi-objective optimization and their demonstrated dependence on the performance of the corresponding EMO procedure made researchers think of parameter-less EMO procedures.

5 NSGA-II and No Additional Parameters

Next, we consider the elitist EMO methodologies and begin our discussion with the elitist non-dominated sorting or NSGA-II [8] algorithm.

The requirement of the additional parameter setting (σ_{share}) in NSGA is avoided in the elitist non-dominated sorting GA or NSGA-II [8]. Although the non-dominated sorting approach is used, the NSGA-II procedure is somewhat different from NSGA in the following aspects:

1. NSGA-II eliminates the need of the additional parameter, which NSGA required.
2. NSGA-II uses an elite-preserving strategy, whereas NSGA did not have any elite-preservation mechanism.
3. NSGA-II uses a computationally fast niching strategy which is applied with objective values, instead of decision parameter values used in NSGA.
4. NSGA-II allows a flexible platform to develop different strategies, such as a steady-state EMO, a preference-based EMO, and others, which were not possible with the proportionate selection framework of NSGA.

The determination of the non-dominated solutions does not require any additional parameter. The clever method of combining offspring and parent populations together and choosing the best half of the combined population according to a non-dominated sorting and crowding scheme did not require any additional parameter in implementing the overall elite-preservation scheme. This way, if a parent solution is better than all offspring solutions, it naturally has a higher chance of getting included in the next population, thereby implementing the elite-preservation principle without any extra parameter. Such a scheme has also been used in other single-objective EA methodologies, such as CHC [15] and evolution strategy studies [36].

Next, we discuss how the niching strategy is implemented without requiring any additional niching parameter in NSGA-II. The original NSGA procedure degraded the *rank* assigned to each solution (based on its non-dominated ranking) by the niche count – a measure of crowdedness near the solution in the decision space. This procedure required a niching parameter. The NSGA-II procedure used a completely different niching strategy. Instead of degrading the fitness or assigning a rank to each solution, a two-stage procedure is adopted here. First, the solutions

are chosen based on the non-domination ranking. Second, solutions, belonging to a maximum ranking which could not be completely accepted to keep the size of the population constant, are evaluated for their crowdedness. This two-stage procedure allows a niching procedure to be applied to a non-dominated set of population members. The niching task remains as the one of finding a subset of population members having as few neighbors as possible. Such a niching task is possible to achieve without requiring any niching parameter.

There are two ways such a task can be achieved: (i) a block reduction strategy or (ii) a one-at-a-time pruning strategy. In both approaches, every population member can be assigned a crowding metric value based on the extent of neighboring solutions in the objective space or in the variable space. Thereafter, in the block reduction strategy, the required number of solutions having the largest crowding metric value (meaning more crowding) can be eliminated at once. In the one-at-a-time pruning strategy [31], only one solution having the largest crowding metric value is eliminated and the crowding metric value is recomputed for the remaining solutions. One solution is eliminated again and the procedure is continued till the required number of solutions are eliminated. It is clear that the latter strategy would constitute a better niching operation than the former one, but at the expense of more computations.

Thus it is clear that a two-phase task of fitness assignment and niching operation allow no additional parameters for achieving both the tasks. NSGA-II procedure only requires the standard EA parameters (population size, operator probabilities etc.) to be supplied.

We argue that requirement of no further parameters in NSGA-II is one of the main reasons of its popularity in EMO studies in the recent past. We observe that the recent EMO conference proceedings (EMO-2005 [2]) had a total of 59 papers on multi-objective optimization and 33 papers (about 56%) used NSGA-II directly or used as basis for comparison. Similarly, 19 of 56 papers (about 34%) of EMO-2003 conference proceedings [17] used NSGA-II. A recent recognition of a high rate of citation of the article describing NSGA-II by the ISI Web of Science [3] is another testimony to its success as an efficient multi-objective optimization algorithm.

6 Other EMO Methodologies and Additional Parameters

The strength Pareto EA or SPEA [43] uses a parameter-less fitness assignment scheme based on domination level of each solution. However, the niching operator and the elite-preserving operators involve additional parameters.

SPEA uses a clustering technique for niching to make a one-at-a-time deletion of crowded solutions. In this approach, every non-dominated solution is assumed to reside in a separate cluster. Thereafter, two clusters having the shortest distance between them are merged together and the solution closest to the centroid of the merged cluster is retained and other solutions in the merged cluster are not considered. This procedure is continued as many times as the required attrition in population members. This way more-crowded solutions are deleted and less-crowded solutions get emphasized. Although this niching strategy apparently does not require any additional niching parameter, the extended SPEA or SPEA2 [42] used the distance to the k-th nearest neighbor to cluster solutions and suggested $k = \sqrt{N + N'}$ (N is the EA population size and N' is the archive size). In this sense, the original SPEA procedure used an additional niching parameter but authors suggested a fixed value of $k = 1$ (in the original SPEA) or $k = \sqrt{N + N'}$ in SPEA2. Besides, the one-at-a-time deletion strategy causes SPEA or SPEA2 to have a large computational burden [12], to have a much better distribution of solutions particularly for problems having more than two objectives.

SPEA and SPEA2 also require an archive, the size of which must be carefully chosen to provide an adequate selection pressure to the better solutions (another parameter!). This is because the archive contains the best non-dominated solutions (elites) found thus far in any generation and a

combination of archive and EA population is used for choosing potential parent solutions. Thus, the ratio between the archive size and EA population size sets up a critical selection pressure for the elite solutions. The developers of SPEA suggested to use an archive size which is about 1/5-th to the size of the EA population. Potentially, the archive size is another additional parameter which SPEA or SPEA2 requires the user to set. It is interesting to note both these parameters (k and archive size N') in associated in the proper working of SPEA or SPEA2 and are not the 'desired parameters' from the point of view of a user.

The Pareto-archived evolution strategy or PAES [30] used a pairwise domination-based selection scheme, which does not require any additional parameter. However, the niching approach compartmentalizes the objective space into a fixed number of hyper-boxes, which required an additional niching parameter. In PAES, the number of divisions in each objective or the size of hyper-boxes in each objective is used for this purpose. The PAES procedure attempts to have only one solution in each hyper-box, thereby ensuring a diversity among population members. Thus, a smaller box-size is, in effect, capable of finding more optimized solutions. Since this parameter directly controls the number of optimized solutions found at the end of the optimization run, it can be called as a 'desired parameter'. By fixing the box-size (niching parameter), the user can control the number of optimized solutions. However, the user does not have a direct control of the exact number of non-dominated solutions to be found, as the the box-size parameter is prespecified and it is not known beforehand the extent of the Pareto-optimal set. The elite-preserving operator compared the newly-created child solution with the parent which created it and used a parameter-less acceptance rule based on domination level of two solutions and the number of solutions resident to hyper-boxes of child and parent.

7 Omni-Optimizer and No Additional Parameters

Recently, the author, with the help of his student, suggested an omni-optimizer which is capable of solving four different optimization tasks without any intervention and additional parameter [14]:

1. Problems having single-objective, single optimum,
2. Problems having single-objective, multiple optima,
3. Problems having multi-objective, single optimum for each Pareto-optimal point, and
4. Problems having multi-objective, multiple optimal for each Pareto-optimal point.

The advantage of such an optimization algorithm is that the algorithm degenerates itself to find the necessary optimal solutions (one or more) depending on the supplied objective(s). This way the user needs only to know a single optimization algorithm and may be able to solve different kinds of optimization problems offered by the description of the problem (objectives and constraints).

The algorithm has a generic structure of a multi-objective optimization algorithm (NSGA-II). The domination criterion of comparing two solutions for superiority degenerates to making a real-valued comparison of two solutions in the case of single-objective optimization. Thus, this first feature of the omni-optimizer is parameter-free. For single-objective optimization, the algorithm also degenerates to an elite-preserving operation by first combining both parent and offspring populations and then systematically choosing half the population. Thus, the elite-preserving operator is also parameter-free.

Since this algorithm claims to find multiple optimal solutions simultaneously, the omni-optimizer has an explicit niching operator. Since the algorithm also claims also to solve both single and multi-objective optimization problems, the niching operator is also expected to perform niching in both spaces: (i) variable space and (ii) objective space. The difficulty with such

a requirement is that the niching information in one space can be completely different from that in the other space. Thus, it may be difficult to come up with a combined niching metric to define the extent of crowding in both spaces. If a sharing function based approach [23] is to be used, two different niching parameters must have to be introduced, thereby requiring the user to set two niching parameters. However, the omni-optimizer uses a parameter-less approach which make a hierarchical comparison of the two niching quantities and constitute a viable algorithm. We give a brief description of the procedure below.

First, every solution in a front is compared with its neighbors objective-wise and an objective-space crowding distance (say D_o) (similar to that in NSGA-II [8]) is computed. The extreme solutions are assigned a very large crowding distance to ensure their presence in the population. Thereafter, a variable-space crowding distance (say D_v) is computed by using a similar procedure, but all computations are performed in the variable space. Here, the extreme solutions in the variable space are not assigned an arbitrary large crowding distance, instead twice the crowding distance from their nearest solutions are assigned. This way, every solution in the population is assigned two niching quantities computed without using any extra niching parameter: the crowding distance in the objective space and crowding distance in the variable space. To choose a subset of solutions from a non-dominated set, the next task is to use these two measures and decide which all solutions to keep and which all solutions to delete. One way to do this would be to choose a threshold for each space and keep all solutions having the crowding distance greater than the chosen threshold value. But this simple-minded procedure will require two new niching parameters to be set by the user. The omni-optimizer uses a parameter-less procedure, which we discuss next.

First, we compute the front-average of objective-wise and variable-wise crowding distance values. Thereafter, instead of comparing the individual crowding distance values with a threshold of some sort, we compare them with these front-wise average values. If a solution does not have above-front-average crowding distance in both spaces, we de-emphasize the solution by assigning the smallest of the two crowding distance values as its overall crowding distance measure. On the other hand, if a solution has above-average crowding distance value in any of the two spaces, we encourage it by assigning it the larger of the two crowding distance values. This procedure is free from any niching parameter and emphasizes any solution which has above-average crowding distance in any of the two spaces.

In the event of a single-objective problem having multiple, identical optimal function value, all high-performing solutions will have very similar objective values, thereby making the objective-wise crowding distance value small (and more or less identical) for each solution. In this case, solutions are chosen mainly based on the variable-wise crowding distance value. The effect is that solutions residing near different and well-separated optima in the variable space get emphasized in the population, thereby allowing multiple optimal solutions to be remain in the population. Most niching procedures work using a similar principle, but unfortunately using a niching parameter. For example, the sharing function approach can find multiple optimal solutions simultaneously, but requires a sharing parameter to be set properly. By comparing a crowding measure with the population-average crowding measure, solutions can be given differential preference, thereby encouraging niche-formation without the need of any additional niching parameter.

Similarly, if in a multi-objective optimization problem, each Pareto-optimal solution corresponds to multiple variable vectors, it would be desirable to locate all or as many such optimal variable vectors as possible. The above two-space crowding distance consideration allows a nice way to find such multiple optimal solutions as well. To illustrate, we consider the following

two-objective minimization problem:

$$\begin{aligned} \text{Minimize} \quad & f_1(x) = \sum_{i=1}^{n} \sin(\pi x_i), \\ \text{Minimize} \quad & f_2(x) = \sum_{i=1}^{n} \cos(\pi x_i), \\ & 0 \le x_i \le 6, \quad i = 1, 2, \ldots, n. \end{aligned} \tag{2}$$

Here, $n = 5$ and both objectives are periodic functions with period of 2. The Pareto-optimal solutions are $x_i \in [2m+1, 2m+3/2]$, where m is an integer. In the above bound, there are exactly three solutions (in [1.0,1.5], [3.0,3.5] and [5.0,5,5]) which produce the same objective values. For every efficient point in the objective space there are in general $3^5 = 243$ Pareto-optimal solutions in the variable space. We choose a population of size 1,000 and run the algorithm for 500 generations to capture as many Pareto-optimal solutions as possible. It is interesting to note that both omni-optimizer and the original NSGA-II find the entire range of efficient points in the objective space, as shown in Figures 3 and 4. However, the variable space plots show a different scenario. The lower diagonal plots in Figure 5 show the performance of the omni-optimizer and the upper diagonal plots show that of the original NSGA-II. It is clear that in a pair-wise plot of variables, all 3^2 or 9 optimal regions are found by the omni-optimizer, whereas since no variable-space crowding is considered in the original NSGA-II, not all optimal combinations are found.

8 Conclusions

In this paper, we have functionally decomposed the tasks needed in an evolutionary multi-objective optimization (EMO) algorithm and identified the critical components which are vulnerable to be designed for additional parameters. These are the niching operator and the elite-preserving operator. By systematically analyzing a number of existing EMO methodologies, we have shown that an EMO methodology can be designed without introducing any additional parameter to those needed in a standard single-objective EA. The elitist non-dominated sorting GA or NSGA-II is one such EMO algorithm and a major reason for its popularity can be contributed to the fact that it does not demand setting any further parameters than those required by single-objective EAs. Often, such parameter-less implementations are innovative and attempt to design such procedures may spur novel ideas in other similar algorithm developmental tasks.

Acknowledgments

The author wishes to thank Santosh Tiwari for implementing the idea of omni-optimizer. The author also acknowledges the support provided by STMicroelectronics for performing this study.

References

[1] T. P. Bagchi and K. Deb. Calibration of GA parameters: The design of experiments approach. *Computer Science and Informatics*, 26(4):46–56, 1996.

[2] C. A. Coello Coello, A. H. Aguirre, and E. Zitzler, editors. *Evolutionary Multi-Criterion Optimization: Third International Conference*. Berlin, Germany: Springer, 2005. LNCS 3410.

[3] K. Deb. IEEE TEC Paper (2002, vol. 6, pp. 182–197) on NSGA-II: Fast breaking paper in the field of engineering. ISI Web of Science, http://esi-topics.com/fbp/2004/february04-KalyanmoyDeb.html.

[4] K. Deb. Multi-objective evolutionary optimization: Past, present and future. In I. C. Parmee, editor, *Evolutionary Design and Manufacture*, pages 225–236. London: Springer, 2000.

[5] K. Deb. *Multi-objective optimization using evolutionary algorithms*. Chichester, UK: Wiley, 2001.

[6] K. Deb and R. B. Agrawal. Simulated binary crossover for continuous search space. *Complex Systems*, 9(2):115–148, 1995.

[7] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *Proceedings of the Parallel Problem Solving from Nature VI (PPSN-VI)*, pages 849–858, 2000.

[8] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[9] K. Deb, A. Anand, and D. Joshi. A computationally efficient evolutionary algorithm for real-parameter optimization. *Evolutionary Computation Journal*, 10(4):371–395, 2002.

[10] K. Deb, S. Chaudhuri, and K. Meitinnen. I-EMO: An interactive evolutionary multi-objective optimization tool. Technical Report KanGAL Report Number 2005005, Department of Mechanical Engineering, IIT Kanpur, India, 2005. Also in Proceedings of Pattern Recognition in Machine Intelligence (PReMI'05), Springer.

[11] K. Deb and D. E. Goldberg. An investigation of niche and species formation in genetic function optimization. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 42–50, 1989.

[12] K. Deb, M. Mohan, and S. Mishra. Towards a quick computation of well-spread pareto-optimal solutions. In *Proceedings of the Second Evolutionary Multi-Criterion Optimization (EMO-03) Conference (LNCS 2632)*, pages 222–236, 2003.

[13] K. Deb and J. Sundar. Reference point based multi-objective optimization using evolutionary algorithms. In *Proceeedings of the Genetic and Evolutionary Computation Conference (GECCO-2006)*, in press.

[14] K. Deb and S. Tiwari. Omni-optimizer: A generic evolutionary algorithm for global optimization. *European Journal of Operations Research (EJOR)*, in press.

[15] L. J. Eshelman. The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. In *Foundations of Genetic Algorithms 1 (FOGA-1)*, pages 265–283, 1991.

[16] L. J. Eshelman and J. D. Schaffer. Real-coded genetic algorithms and interval-schemata. In *Foundations of Genetic Algorithms 2 (FOGA-2)*, pages 187–202, 1993.

[17] C. Fonseca, P. Fleming, E. Zitzler, K. Deb, and L. Thiele. *Proceedings of the Second Evolutionary Multi-Criterion Optimization (EMO-03) Conference (Lecture Notes in Computer Science (LNCS) 2632)*. Heidelberg: Springer, 2003.

[18] C. M. Fonseca and P. J. Fleming. An overview of evolutionary algorithms in multi-objective optimization. *Evolutionary Computation Journal*, 3(1):1–16, 1995.

[19] D. E. Goldberg. *Genetic Algorithms for Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[20] D. E. Goldberg and K. Deb. A comparison of selection schemes used in genetic algorithms. In *Foundations of Genetic Algorithms 1 (FOGA-1)*, pages 69–93, 1991.

[21] D. E. Goldberg, K. Deb, and J. H. Clark. Genetic algorithms, noise, and the sizing of populations. *Complex Systems*, 6(4):333–362, 1992.

[22] D. E. Goldberg, K. Deb, and D. Thierens. Toward a better understanding of mixing in genetic algorithms. *Journal of the Society of Instruments and Control Engineers (SICE)*, 32(1):10–16, 1993.

[23] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, pages 41–49, 1987.

[24] G. Harik, E. Cantú-Paz, D. E. Goldberg, and B. L. Miller. The gambler's ruin problem, genetic algorithms, and the sizing of populations. *Evolutionary Computation Journal*, 7(3):231–254, 1999.

[25] T. Higuchi, S. Tsutsui, and M. Yamamura. Theoretical analysis of simplex crossover for real-coded genetic algorithms. In *Parallel Problem Solving from Nature (PPSN-VI)*, pages 365–374, 2000.

[26] J. Horn, N. Nafploitis, and D.E. Goldberg. A niched Pareto genetic algorithm for multiobjective optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation*, pages 82–87, 1994.

[27] K.A. De Jong and J. Sharma. Generation gap revisited. In *Foundations of Genetic Algorithms (FOGA-II)*, pages 19–28, 1992.

[28] James Kennedy and Russell C. Eberhart. *Swarm intelligence*. Morgan Kaufmann, 2001.

[29] H. Kita, I. Ono, and S. Kobayashi. Multi-parental extension of the unimodal normal distribution crossover for real-coded genetic algorithms. In *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 1581–1587, 1999.

[30] Joshua D. Knowles and David W. Corne. Approximating the non-dominated front using the Pareto archived evolution strategy. *Evolutionary Computation Journal*, 8(2):149–172, 2000.

[31] S. Kukkonen and K. Deb. Improved pruning of non-dominated solutions based on crowding distance for bi-objective optimization problems. In *Proceedings of the Congress on Evolutionary Computation (CEC-06)*, in press.

[32] N. J. Radcliffe. Forma analysis and random respectful recombination. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 222–229, 1991.

[33] C. R. Reeves. Using genetic algorithms with small populations. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 92–99, 1993.

[34] H. Satoh, M. Yamamura, and S. Kobayashi. Minimal generation gap model for gas considering both exploration and exploitation. In *Proceedings of the IIZUKA: Methodologies for the Conception, Design, and Application of Intelligent Systems*, pages 494–497, 1996.

[35] J. D. Schaffer, R. A. Caruana, L. J. Eshelman, and R. Das. A study of control parameters affecting online performance of genetic algorithms. In *Proceedings of the International Conference on Genetic Algorithms*, pages 51–60, 1989.

[36] H.-P. Schwefel. *Evolution and Optimum Seeking*. New York: Wiley, 1995.

[37] N. Srinivas and K. Deb. Multi-objective function optimization using non-dominated sorting genetic algorithms. *Evolutionary Computation Journal*, 2(3):221–248, 1994.

[38] R. Storn and K. Price. Differential evolution – A fast and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341–359, 1997.

[39] H.-M. Voigt, H. Mühlenbein, and D. Cvetković. Fuzzy recombination for the Breeder Genetic Algorithm. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 104–111, 1995.

[40] I. Wegener and T. Jansen. Real royal road functions - where crossover provably is essential. *Discrete Applied Mathematics*, 149:111–125, 2005.

[41] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1977.

[42] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In K. C. Giannakoglou, D. T. Tsahalis, J. Périaux, K. D. Papailiou, and T. Fogarty, editors, *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, pages 95–100, Athens, Greece, 2001. International Center for Numerical Methods in Engineering (Cmine).

[43] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.

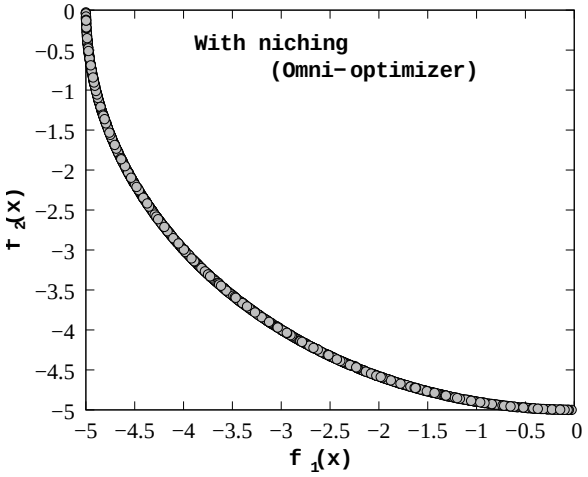

Figure 3: Efficient points using omni-optimizer.

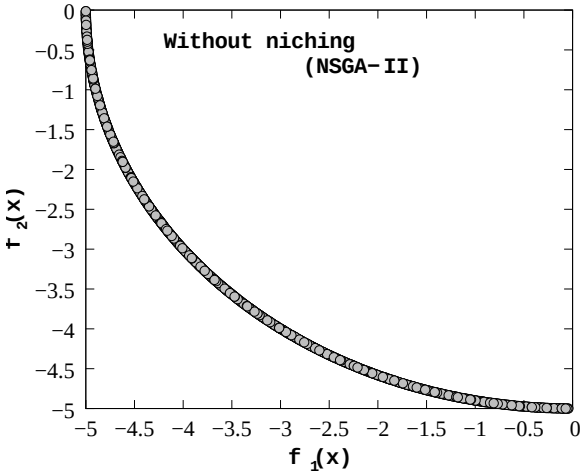

Figure 4: Efficient points using NSGA-II.

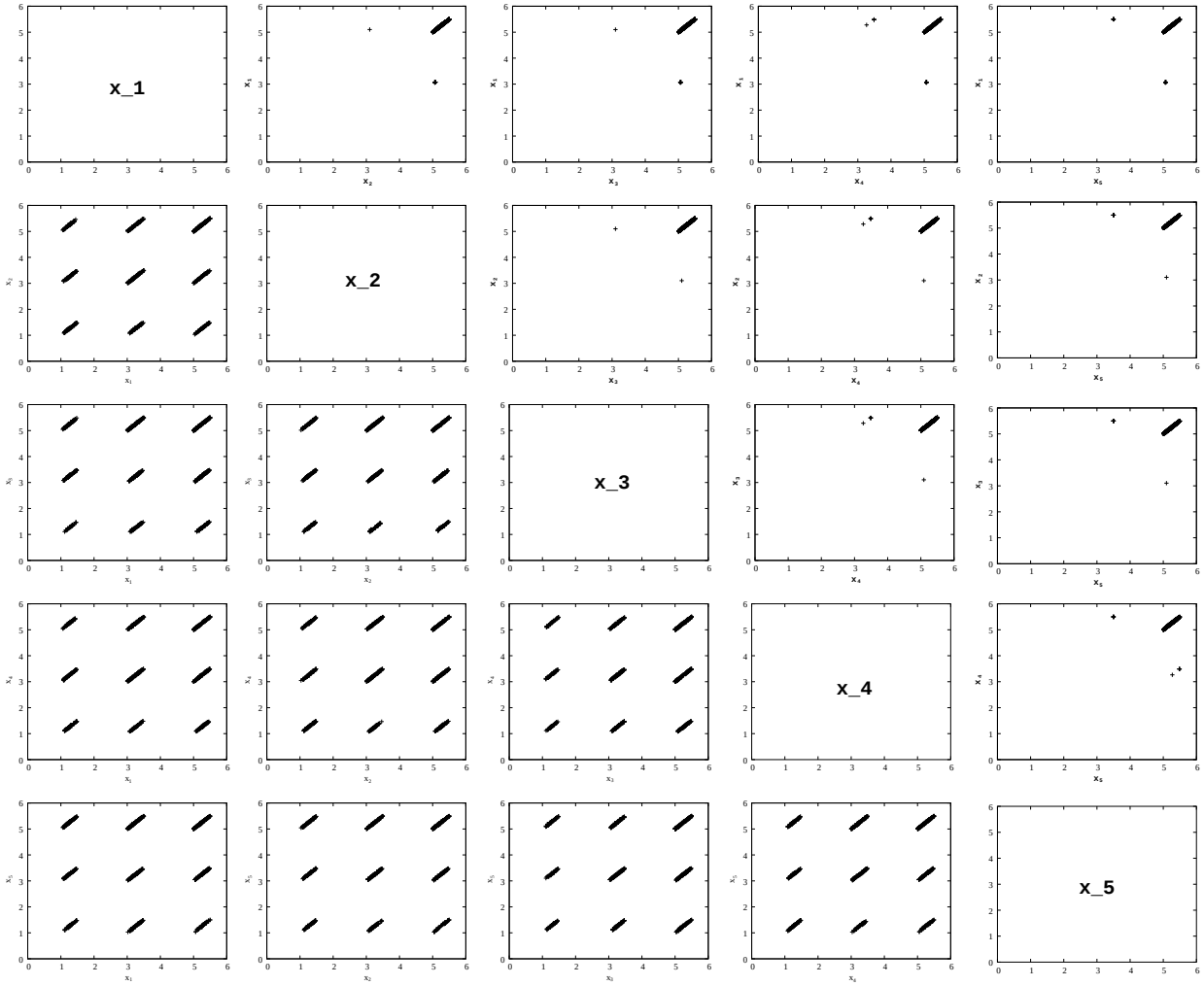

Figure 5: Pareto-optimal solutions with omni-optimizer (left) and NSGA-II (right). The axes in a (i, j)-plot correspond to variables x_i and x_j.