CHAPTER 10

Pareto Multiobjective Optimization

PATRICK N. NGATCHOU, ANAHITA ZAREI, WARREN L.J. FOX, and MOHAMED A. EL-SHARKAWI

10.1 INTRODUCTION

Compared with single-objective (SO) optimization problems, which have a unique solution, the solution to multiobjective (MO) problems consists of sets of trade-offs between objectives. The goal of multiobjective optimization (MOO) algorithms is to generate these trade-offs [1–4]. Exploring all these trade-offs is particularly important because it provides the system designer/operator with the ability to understand and weigh the different choices available to them.

Solving MO problems has traditionally consisted of converting all objectives into a SO function. The ultimate goal is to find the solution that minimizes or maximizes this single objective while maintaining the physical constraints of the system or process. The optimization solution results in a single value that reflects a compromise between all objectives. This simple optimization process is no longer acceptable for systems with multiple conflicting objectives: System engineers may desire to know all possible optimized solutions of all objectives simultaneously. In the business world, it is known as a trade-off analysis.

It is commonplace for real-world optimization problems in business, management, and engineering to feature multiple, generally conflicting objectives. For instance, many businesses face the problem of minimizing their operating cost while maintaining a stable work force. A common problem in the integrated circuit manufacturing industry is placing the largest number of functional blocks on a chip while minimizing the area and/or power dissipation.

The planning and operation of power systems inherently requires solving multiobjective optimization problems. For instance, in environmental/economic load dispatch, the problem consists of minimizing operation cost while minimizing fossil fuel emissions and system losses [4]. Also, when designing transmission networks, the

Modern Heuristic Optimization Techniques. Edited by K. Y. Lee and M. A. El-Sharkawi

optimization of reactive resources location and sizing of the transmission and distribution system is another example of a MO problem [6, 7].

This chapter focuses on heuristic multiobjective optimization, particularly with population-based stochastic algorithms such as evolutionary algorithms. The next section presents the basic principles behind MOO, notably introducing the Pareto optimality concepts and formulation. This is followed in Section 10.3 with a presentation of the different solution approaches. A distinction is made between classic and modern techniques. The former consist of converting the MO problem into an SO problem, whereas the latter take advantage of population-based meta-heuristics to explore the entire trade-off curve. Section 10.4 addresses the problem of performance evaluation of MO optimizers.

10.2 BASIC PRINCIPLES

For illustration purposes, consider the hypothetical problem of determining, given a choice of transportation means, the most efficient of them based on distance covered in a day and energy used in the process. In a MOO framework, the two objectives are maximization of distance (or equivalently, minimization of $1/\text{distance}$) and minimization of energy. The transportation modes considered are walking, bicycle, cow, car, motorcycle, horse, airplane, rocket, balloon, boat, and scooter. Figure 10.1 shows a plot of $1/\text{distance}$ versus consumed energy for these transportation modes. Given the same amount of food, it would be generally expected that a cow would cover a shorter distance than a horse. Also, a rocket covers a longer distance than a plane

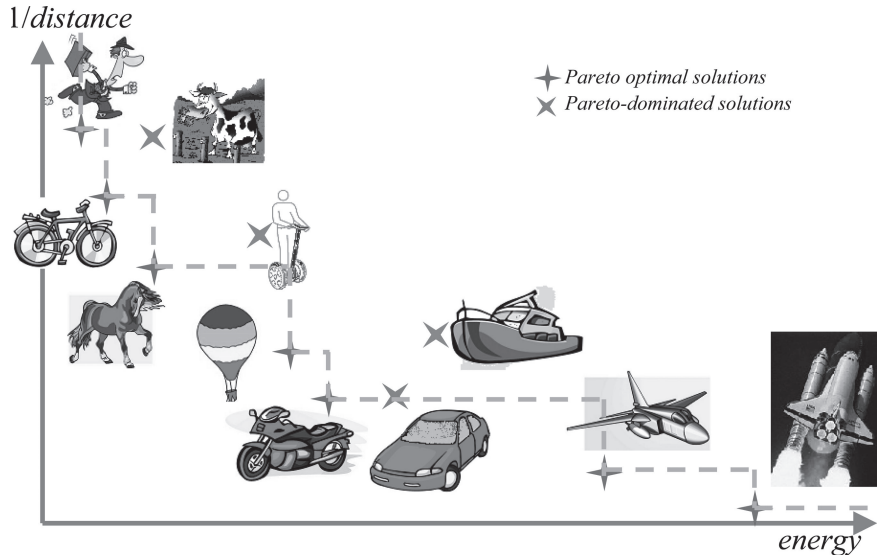

FIGURE 10.1 Biobjective optimization of distance and energy.

while consuming a higher amount of energy. The points on the dotted line represent a set of optimum solutions. Sections 10.2.1 and 10.2.2 will refer to this example.

10.2.1 Generic Formulation of MO Problems

The general MO problem requiring the optimization of N objectives may be formulated as follows:

$$\begin{array}{ll} \text{Minimize} & \vec{\mathbf{y}} = \vec{\mathbf{F}}(\vec{\mathbf{x}}) = [f_1(\vec{\mathbf{x}}), f_2(\vec{\mathbf{x}}), \ldots, f_N(\vec{\mathbf{x}})]^T \\ \text{subject to} & g_j(\vec{\mathbf{x}}) \leq 0, j = 1, 2, \ldots, M, \end{array} \tag{10.1}$$

where $\vec{\mathbf{x}} \in \Omega$.

In Eq. (10.1), $\vec{\mathbf{y}}$ is the objective vector, the g_j's represent the constraints, and $\vec{\mathbf{x}}$ is the decision vector representing the decision variables within a parameter space Ω. The space spanned by the objective vectors is called the objective space. The subspace of the objective vectors that satisfies the constraints is called the feasible space.

The *utopian* solution is the solution that is optimal for all objectives.

$$\begin{array}{l} \vec{\mathbf{x}}_0{}^* \in \Omega : \forall \vec{x} \in \Omega, f_i(\vec{\mathbf{x}}_0{}^*) \leq f_i(\vec{\mathbf{x}}) \\ \text{for } i \in \{1, 2, \ldots, N\}. \end{array} \tag{10.2}$$

For $N = 1$, the MO problem is reduced to an SO problem. In that case, the utopian solution is simply the global optimum. It always exists, even if it cannot be found.

For the more general case where $N > 1$, the utopian solution does not generally exist because the individual objective functions are typically conflicting. Rather, there is a possibly uncountable set of solutions that represent different compromises or trade-offs between the objectives.

In the earlier example, $N = 2$ and the parameter space Ω is the set of transportation modes. The objective vector would then be:

$$\vec{\mathbf{y}} = [f_1(x), f_2(x)] = [1/\text{distance}(x), \text{energy}(x)], x \in \Omega.$$

Maximization of distance and minimization of energy are two conflicting objectives in this example. Therefore, instead of a single solution, there exists a set of solutions that appear on the dotted line in Figure 10.1.

10.2.2 Pareto Optimality Concepts

To compare candidate solutions to the MO problems, the concepts of Pareto dominance and Pareto optimality are commonly used. These concepts were originally introduced by Francis Ysidro and then generalized by Vilfredo Pareto [3].

A solution belongs to the Pareto set if there is no other solution that can improve at least one of the objectives without degrading any other objective. Formally, a

decision vector $\vec{\mathbf{u}} \in \Omega$ is said to *Pareto-dominate* vector $\vec{\mathbf{v}} \in \Omega$, in a minimization context, if and only if:

$$\begin{aligned} &\forall i \in \{1, \text{K}, N\}, f_i(\vec{\mathbf{u}}) \leq f_i(\vec{\mathbf{v}}), \\ &\text{and } \exists j \in \{1, \text{K}, N\}, f_i(\vec{\mathbf{u}}) < f_i(\vec{\mathbf{v}}). \end{aligned} \tag{10.3}$$

In the context of MOO, Pareto dominance is used to compare and rank decision vectors: $\vec{\mathbf{u}}$ dominating $\vec{\mathbf{v}}$ in the Pareto sense means that $\vec{\mathbf{F}}(\vec{\mathbf{u}})$ is either better than or the same as $\vec{\mathbf{F}}(\vec{\mathbf{v}})$ for all objectives, and there is at least one objective function for which $\vec{\mathbf{F}}(\vec{\mathbf{u}})$ is strictly better than $\vec{\mathbf{F}}(\vec{\mathbf{v}})$. For instance, in Figure 10.1, a motorcycle Pareto dominates a car, because $f_1(\text{motorcycle}) = f_2(\text{car})$ (a motorcycle travels the same distance as a car), and $f_1(\text{motorcycle}) < f_2(\text{car})$ (a motorcycle consumes less energy than a car).

A solution $\vec{\mathbf{a}}$ is said to be Pareto optimal if and only if there does not exist another solution that dominates it. In other words, solution $\vec{\mathbf{a}}$ cannot be improved in one of the objectives without adversely affecting at least one other objective. The corresponding objective vector $\vec{\mathbf{F}}(\vec{\mathbf{a}})$ is called a Pareto dominant vector, or noninferior or nondominated vector. The set of all Pareto optimal solutions is called the Pareto optimal set. The corresponding objective vectors are said to be on the Pareto front. It is generally impossible to come up with an analytical expression of the Pareto front.

On Fig. 10.1, the Pareto optimal set consists of points on the dotted line. The boat, scooter, and cow do not belong to the Pareto optimal set. For instance, the scooter cannot belong to the Pareto optimal set due to the presence of another element, the horse, which improves both objectives by traveling a longer distance and consuming less energy. Similarly, the bicycle and horse both Pareto-dominate the cow, and the balloon, motorcycle, and car Pareto-dominate the boat.

In general, given a set of vectors in objective space, it is possible to determine the set of vectors corresponding with nondominated solutions by applying Pareto dominance comparison. Figure 10.2 depicts Pareto fronts for biobjective minimization and maximization problems.

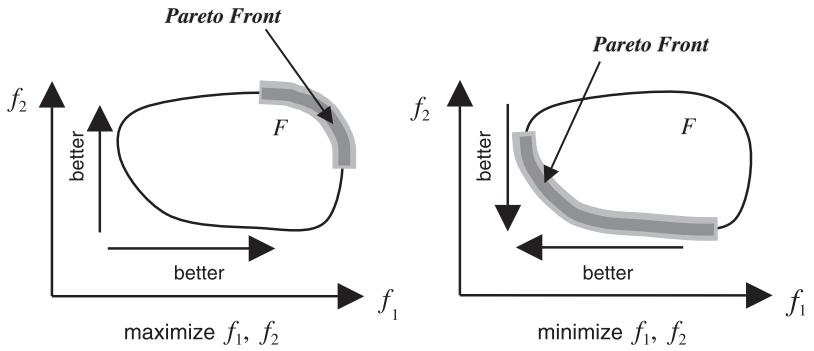

FIGURE 10.2 Illustration of Pareto front for a biobjective optimization problem.

10.2.3 Objectives of Multiobjective Optimization

MOO consists of determining all solutions to the MO problem that are optimal in the Pareto sense. In contrast with SO problems where the solution consists of a single element of the search space, the solution to MO problems consists of a set of elements of the search space. The objective of MOO is to determine the best approximation to this Pareto optimal set.

For stochastic optimizers in particular, solutions to MO problems are not always identical, and comparing them is often difficult because many criteria must be taken into account (Fig. 10.3). Solving MO problems is itself a multiobjective problem. The objectives are to

(a) Minimize the distance between the approximation set generated by the algorithm and the Pareto front;
(b) Ensure a good distribution of solutions along the approximation set (uniform if possible);
(c) Maximize the range covered by solutions along each of the objectives.

The above requirements guide the design of MOO algorithms (Section 10.3), the evaluation of their performance, and the quality of the results they generate (Section 10.4).

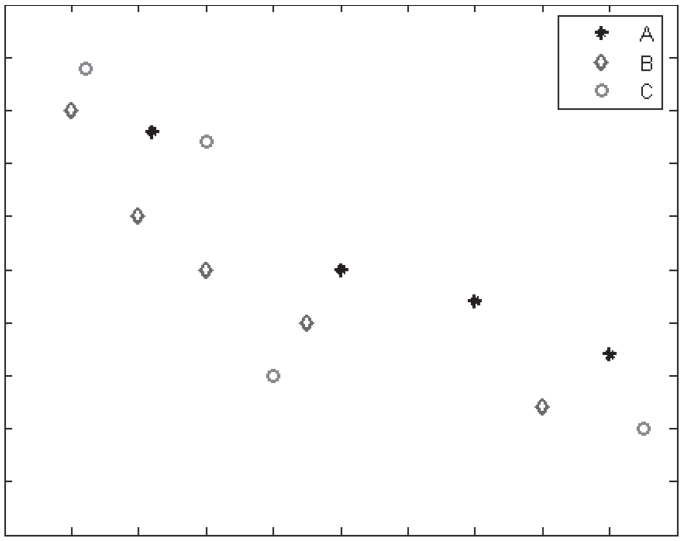

FIGURE 10.3 Illustration of the difficulty in comparing solutions to MO problems and motivating the objectives of MOO. In the objective space, we represent three hypothetical solutions of a biobjective minimization problem. The distribution of trade-off points in set *A* is fairly uniform, but the objective space coverage range is not good as for sets *B* and *C*. Set *C* covers the largest range but does not have a good distribution. Set *B* falls somewhere in between sets *A* and *C*. A good MOO Algorithm will try to generate an approximation set with a uniform distribution of trade-off points that at the same time covers the largest range in the objective space and is very close to the Pareto front.

10.3 SOLUTION APPROACHES

Solving MO problems has evolved over the years, and techniques or algorithms can be categorized along chronological lines. Classic approaches, which have roots in the operations research and optimization theory fields, essentially consist of converting the MO problem into a SO problem, which then can be solved using traditional scalar optimization techniques.

Classic approaches traditionally used mathematical programming (e.g., steepest descent, Newton–Raphson) to solve the SO problem. However these approaches are ill-suited for real-world problems where mixed-type decision variables, constraints, or nondifferentiable objective functions are all too common. It is only with the development of meta-heuristic algorithms and a parallel increase in computational power that these became incorporated into optimizers to tackle real-world problems.

The use of population-based meta-heuristics characterizes the second class of MO solvers, which is historically more recent. Indeed, population-based algorithms such as evolutionary algorithms, particle swarm optimization, or ant colony optimization allow direct generation of trade-off curves in a single run. Hence, the modern techniques are geared toward direct determination of the Pareto front by optimizing all the objectives concurrently.

10.3.1 Classic Methods

Classic methods were essentially techniques developed by the operations research community to address the problem of multicriteria decision making (MCDM). Given multiple objectives and preferential information about these objectives, the MO problem is converted into an SO problem by either aggregating the objective functions or optimizing the most important objective and treating the others as constraints. The SO problem can then be solved using traditional scalar-valued optimization techniques. These techniques are geared toward finding a single solution and are ideal for cases where preferential information about the objectives is known *a priori*. Using either ranking the objectives in order of importance, or target optimal values for each objective, the goal is to find the solution that best satisfies the criteria and additional information (preferences) provided by the Decision Maker (DM). In order to generate a trade-off curve, the solution procedure has to be reapplied after modifying the aggregation modalities or design criteria.

In the general case, and in order to generate an approximation to the nondominated front, all that is needed is to modify the aggregation parameters and solve the newly created SO problem. Below are some examples.

10.3.1.1 Weighted Aggregation A simple and still very popular method is the weighted aggregation method. This is a special case of the utility function method, which converts the MO problem into an SO problem by applying a function operator to the objective vector. This function is designed by the DM to capture their preferences [2]. The utility function used in the case of weighted aggregation is a

linear combination of the objectives:

$$\text{Minimize } Z = \sum_{j=1}^{N} w_j f_j(\vec{\mathbf{x}}) \quad \text{with } w_j \geq 0 \text{ and } \sum_{j=1}^{N} w_j = 1, \tag{10.4}$$

where the weights (w_j's) are chosen to reflect the relative importance the DM attaches to the objectives and must be specified for each of the N objectives *a priori*. Solving equation (10.4) yields a single solution that represents the best compromise given the chosen weights. To obtain an approximation set, the search has to be repeated with different values for the weights, which is computationally efficient.

Conventional weighted aggregation misses concave portions of the Pareto front. To alleviate this problem, a variant of this method called dynamic weighted aggregation (DWA) was developed. In DWA, the weights are incrementally and systematically modified using periodic functions [8, 9]. This method is unfortunately limited to biobjective problems.

10.3.1.2 Goal Programming A closely related variation of the weighted aggregation technique is the goal programming or goal attainment technique. This method seeks to minimize deviation from prespecified goals. Equation (10.5) is a common formulation:

$$\text{Minimize } Z = \sum_{j=1}^{N} w_j |f_j(\vec{\mathbf{x}}_\mathbf{j}) - T_j|, \tag{10.5}$$

where T_j represents the target or goal set by the DM for the jth objective function, and the w_j's now capture the priorities [2]. As in the weighted aggregation approach, the main drawback of this approach is the need for *a priori* information.

10.3.1.3 ε-Constraint This is a method designed to discover Pareto optimal solutions based on optimization of one objective while treating the other objectives as constraints bound by some allowable range ε_i. The problem is repeatedly solved for different values of ε_i to generate the entire Pareto set:

Step 1: Minimize $f_k(\vec{\mathbf{x}})$, $\vec{\mathbf{x}} \in \Omega$ subject to $f_i(\vec{\mathbf{x}}) \leq \varepsilon_i$ and $g_j(\vec{\mathbf{x}}) \leq 0$; $i = 1, \text{K}, N$ $(i \neq k)$, and $j = 1, \text{K}, M$

Step 2: Repeat step 1 for different values of ε_i.

This is a relatively simple technique, yet it is computationally intensive. Furthermore, the solutions found are not necessarily globally nondominated [10].

10.3.1.4 Discussion on Classic Methods Classic methods attempt to ease the decision-making process by incorporating *a priori* preferential information

from the DM and are geared toward finding the single solution representing the best compromise solution. To examine all possible trade-offs requires systematic variation of the aggregation parameters before solving the problem, which makes the approach inefficient although simple to implement. There are further limitations: It is difficult to control the diversity of solutions in the approximation set, and more importantly, the techniques are sensitive to the shape of the global Pareto front [3, 10].

10.3.2 Intelligent Methods

10.3.2.1 Background Meta-heuristic algorithms are increasingly used to tackle optimization problems. Meta-heuristic algorithms are successful SO solvers because they allow handling of real-world optimization problems for which, by virtue of their complexity (ill-defined function, nondifferentiability, etc.), it is impossible to find exact algorithms. For these types of problems, meta-heuristics are a practical way to generate acceptable solutions, even though they cannot guarantee optimality. Another advantage is the ability to incorporate problem-specific knowledge to improve the quality of the solutions. In the context of MOO, they offer greater flexibility for the DM, mainly in cases where no *a priori* information is available as is the case for most real-world MO problems. Hence, they were first hybridized with the mathematical programming techniques in the classic solvers. Several variants of meta-heuristic algorithms exist: single-point stochastic search algorithms such as simulated annealing, or population-based algorithms such as evolutionary algorithms of which genetic algorithms are a special class (Fig. 10.4).

The techniques presented here are mainly population-based algorithms, which are set apart by their ability to generate an approximation set in a single run. In population-based algorithms, several candidate solutions are evaluated in a single iteration. This is the characteristic that makes them suitable for MOO and mainly define the techniques presented in this section. When applied to MO problems, population-based algorithms can generate an approximation of the Pareto front in a single iteration [3, 11–13].

10.3.2.2 Structure of Population-Based MOO Solvers The general structure of EA-based MO solvers is similar to the one used for SOO (Fig. 10.5). The different steps are now adapted so as to meet the objectives of MOO that were outlined in Section 10.2.3. *Fitness assignment* controls convergence (i.e., how to guide the population to nondominated solutions). To prevent premature convergence to a region of the front, *diversity* mechanisms such as niching are included in the determination of an individual's fitness. Diversity also allows one to control the spread of the approximation set. Finally, a form of *elitism* is applied to prevent the deterioration problem whereby nondominated solutions may disappear from one generation to the next. Once fitness of individuals is computed, selection and variation can be performed.

10.3.2.2.1 Fitness Assignment There are three methods of fitness assignment: aggregation-based, criterion-based, and Pareto-based. Aggregation-based assignment

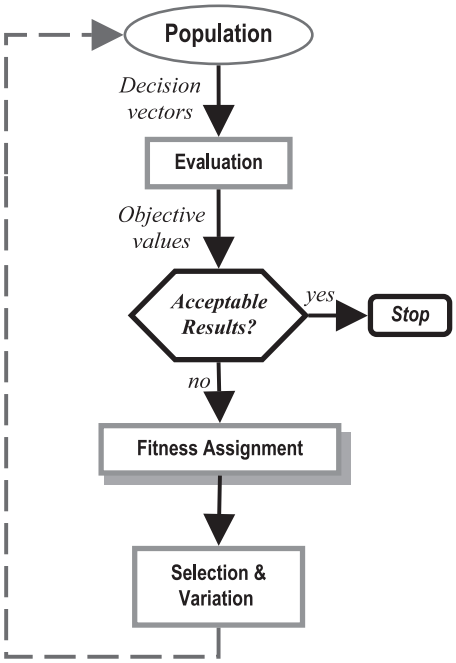

FIGURE 10.4 Generic structure of evolutionary algorithms. Evolutionary computing emulates the biological evolution process. A population of individuals representing different solutions is iteratively evolving to find the optimal solutions. At each iteration, the fittest individuals are chosen, and then variation operations (mutation and crossover) are applied, thus yielding a new generation (offspring).

consists in evaluating the fitness of each individual based on a weighted aggregation of the objectives. In this sense, it is an extension of the classic weighted aggregation method presented in Section 10.3.1. An early implementation of this technique is Haleja & Lin's genetic algorithm (HLGA). To explore the different parts of the Pareto front, they apply systematic variation of the aggregation weights [11].

An example of criterion-based assignment is Schaffer's vector-evaluated genetic algorithm (VEGA) [14], which is a non–Pareto-based technique that differs from the conventional genetic algorithm only in the way in which the selection step is performed. At each generation, the population is divided into as many equal-size subgroups as there are objectives, and the fittest individuals for each objective function are selected (Fig. 10.6).

The VEGA algorithm is easy to implement. However, it suffers from the speciation problem (i.e., evolution of species that excel in one of the objectives). This causes the algorithm to fail to generate compromise solutions (i.e., solutions that are not necessarily the best in one objective but are optimal in the Pareto sense). In addition, the algorithm is sensitive to the shape of the Pareto front.

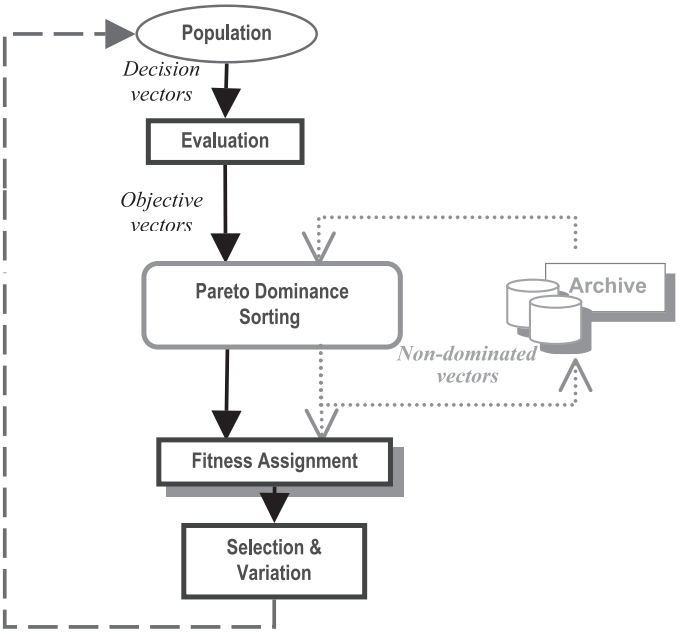

FIGURE 10.5 Pareto front generation using population-based techniques.

Pareto-based fitness assignment is the most popular and efficient technique. Here, Pareto-dominance is explicitly applied in order to determine the probability of replication of an individual. Given a population, a simple method is to find the set of nondominated individuals. These are assigned the highest rank and eliminated from further contention. The process is then repeated with the remaining individuals until the entire population is ranked and assigned a fitness value proportional to the

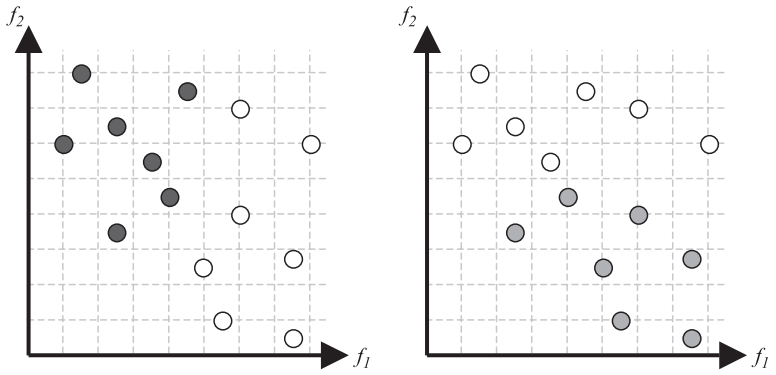

FIGURE 10.6 Illustration of VEGA approach for a biobjective minimization problem. A population of 14 candidate solutions is represented in object space. On the left, the individuals selected according to f_1; on the right, individuals selected according to f_2.

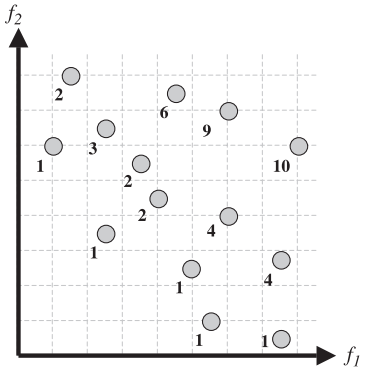

FIGURE 10.7 Illustration of fitness computation for MOGA in a biobjective minimization problem. The rank of a given individual corresponds with 1 plus the number of individuals by which it is dominated. Nondominated individuals have rank 1. The number by each individual is their rank.

ranks. The *multiobjective genetic algorithm* (MOGA) is an algorithm implementing Pareto-based fitness assignment [15]. Each individual is assigned a rank equal to 1 plus the number of individuals by which that individual is dominated. Hence, nondominated individuals have rank equal to 1. Fitness values are then assigned to the ranked population with the lowest rank having the highest fitness (Fig. 10.7).

10.3.2.2.2 Diversity In conjunction with fitness assignment mechanism, an appropriate niching mechanism is necessary to prevent the algorithm from converging to a single region of the Pareto front [10]. Niching methods are inspired from statistical sampling techniques [11]. A popular niching technique called sharing consists of regulating the density of solutions in the hyperspace spanned by either the objective vector or the decision vector. Sharing is often used in the computation of the fitness value. For example, in the MOGA algorithm discussed earlier, an objective space density-based fitness sharing is applied after population ranking. This allows differentiating individuals having identical ranks and favoring those that are in sparsely occupied regions of the objective space.

10.3.2.2.3 Elitism In EA-based solvers, an elitist strategy refers to a mechanism by which the fittest individuals found during the evolutionary search are always copied to the next generation. This increases exploitation, helps convergence, and allows the algorithm to concurrently search multiple local optima.

In MOO, elitism has been proved to be a requirement for convergence to the true Pareto front. The idea is that at each generation, the nondominated solutions found in the population are compared with an external set of nondominated solutions found throughout the entire search process. One of the early algorithms to explicitly incorporate elitism is the *strength pareto evolutionary algorithm* (SPEA) originally proposed in [16]. In SPEA, a repository or external archive is used to maintain

nondominated solutions and is updated at each generation if better nondominated solutions are found. In modern MOO algorithms, the use of an external repository has become standard.

Elitism is intertwined with the diversity and fitness assignment mechanisms. If using a finite-size external repository, diversity criteria govern the insertion/deletion of nondominated solutions in the full archive. Also, elements from this archive may be used in fitness assignment and selection to create the next generation of individuals.

10.3.2.3 Common Population-Based MO Algorithms This section presents some popular population-based MO algorithms. Most of them are based on EAs, but others are inspired from swarm intelligence. The list is by no means exhaustive and is not claimed to be the state-of-the-art MO algorithms. However, it showcases the use of different approaches for incorporating fitness assignment, diversity, and elitism.

The *nondominated sorting genetic algorithm* (NSGA) uses a layered classification technique [17]. All nondominated individuals are assigned the same fitness value and sharing is applied in the decision variable space. The process is repeated for the remainder of the population with a progressively lower fitness value assigned to the nondominated individuals. Because of the iterative ranking process, NSGA is computationally expensive. Furthermore, it does not have any elitism mechanism. NSGA-II, an improved version of NSGA, was introduced to address these issues. The elitist strategy employed in NSGA-II is peculiar in that it does not involve an external repository. Rather, the best offspring and best parents are combined. It uses nondominated ranking where the number of elements a particular solution dominates and is dominated by is taken into account, as well as a local neighborhood crowding for diversity [18] (Fig. 10.8).

In the *niched Pareto genetic algorithm* (NPGA) [19], instead of bilateral direct comparison, two individuals are compared with respect to a comparison set

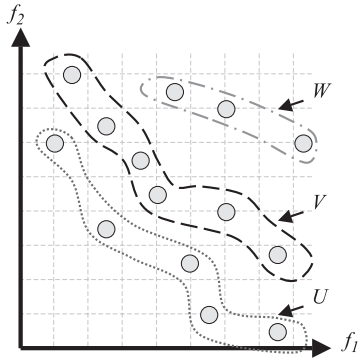

FIGURE 10.8 Illustration of fitness computation for NSGA in a biobjective minimization problem. A layered classification technique is used whereby the population is incrementally sorted using Pareto dominance. Individuals in set *U* have the same fitness value, which is higher than the fitness of individuals in set *V*, which in turn are superior to individuals in set *W*.

(usually 10% of the entire population). When one candidate is dominated by the set while the other is not, the latter is selected. If neither or both the candidates are dominated, fitness sharing is used to decide selection. NPGA introduces a new variable (size of the comparison set) but is computationally faster than the previous techniques as the selection step is applied only to a subset of the population.

As mentioned earlier, in SPEA, elitism is applied through an external archive that not only maintains the nondominated solutions found during the evolutionary search but also seeds the next generation [16]. Fitness values are assigned to both population and archive members. A MOGA-style fitness assignment is applied to archive members: the fitness, or strength, of each archive member is equal to the number of population members that are dominated by it, divided by the population size plus one. As for population members, their fitness is one added to the sum of the strength values of all archive members that dominate it. During selection, population and archive members are merged, and individuals and tournament selection is performed favoring individuals with *lowest* fitness to be selected. Hence, archive members have the highest chance of appearing in the next generation.

SPEA2 was developed to address some of the major drawbacks of SPEA [20]. First, in SPEA, the fitness assignment procedure often results in different individuals having the same fitness. Second, diversity is not explicitly taken into account outside of the archive management. This becomes especially problematic when there are more than two objectives. Finally, SPEA's archive management is such that the size of the archive varies during the optimization, and often solutions at the archive's boundaries are lost. SPEA2 uses a finer-grain fitness assignment procedure that takes into account the number of individuals a population member dominates in addition to the number of archive members that dominates it. Furthermore, a neighborhood distance-based density estimation term is added to the fitness computation. Finally, the archive update is performed so that the number of elements in the archive remains constant, and solutions at the boundaries of the space spanned by the approximation set are maintained.

The *Pareto archive evolution strategy* (PAES) also uses an external archive. The role of the archive is limited to storing the nondominated solutions and providing a source of comparison for ranking candidate solutions. However, archive members are not involved in the mutation and crossover procedures. The interesting feature of this algorithm is in the *adaptive grid* at the heart of its diversity and niching mechanisms. The objective space is divided into a recursive manner, thus creating a multidimensional coordinate system, or grid, over the objective space. A crowding-based fitness sharing mechanism is applied by first determining the location of solutions within this grid and estimating the density of solutions per cell. Individuals corresponding with solutions in the less crowded cells have higher fitness [21].

Other population-based approaches have also been applied to MO problems. Most notable is particle swarm optimization (PSO), a simple and robust optimizer motivated by social behavior such as bird flocking. In PSO, the individuals (called particles) fly around the search space. During their flight, each particle adjusts its position and velocity based on its own experience (personal best) and the group's experience (global best) [22, 23]. Several PSO-based MO solvers have been

proposed. One example is the *multiobjective particle swarm optimizer* (MOPSO). In MOPSO, Pareto-dominance is used to update each particle's personal best. A new global best is selected at each PSO epoch from the set of nondominated solutions using a selection mechanism inspired by PAES's adaptive grid procedure. MOPSO is able to promote exploration of the less densely populated regions of the objective space while maintaining a uniform distribution [24].

10.3.2.4 Discussion on Modern Methods The meta-heuristic techniques successfully address the limitations of the classic approaches. They can generate multiple solutions in a single run, and the optimization can be performed without *a priori* information about the objectives' relative importance. These techniques can handle problems with incommensurable or mixed-type objectives. They are not susceptible to the shape of the Pareto front. Their main drawback is performance degradation as the number of objectives increases, as there do not exist computationally efficient methods to perform Pareto ranking. Furthermore, they require additional parameters such as a sharing factor, or the number of Pareto samples.

10.4 PERFORMANCE ANALYSIS

10.4.1 Objective of Performance Assessment

Performance assessment is important in the development and application of meta-heuristic algorithms. Especially for stochastic optimization algorithms, it is essential to gauge the quality of the solutions generated by the optimizer and also the computational resources necessary to achieve the results. It is also important for tuning the multiple parameters all too common in this class of optimizers or even comparing the quality of two different optimizers for a given class of problems.

In the case of SO optimization, where the goal is to find the decision vector that satisfies a minimization or maximization objective, the typical performance analysis consists of Monte Carlo simulations, statistical characterization of computational intensity (CPU cycles, run time, and/or number of function evaluations), and quality of the solution, found by analyzing the distribution of the output of the solver both in the objective and decision spaces. If the global optima are known, some quantitative metrics are distance to nominal global optima, or average number of iterations necessary to get within a certain range of global optima. These metrics can then be used to compare different optimizers or measure the effect of the different algorithmic parameters.

In the case of MOO, however, a difficulty arises from the fact that the solution is not a point but rather a set of points. Assessing the intrinsic performance of the optimizer requires comparison operations on sets. In the early days of MOO, people relied on qualitative assessment by visual comparison of approximation sets. In recent years, because of the development of several MOO optimizers, it has become important to develop quantitative metrics and comparison methods, as well as reliable test functions, to evaluate the performance of these different optimizers [11, 25].

10.4.2 Comparison Methodologies

Comparison methodologies all involve transformation of the approximation sets generated by an algorithm into another representation. Statistical testing procedures may then be applied on the output representation to compare the algorithm. This section will present an outline of comparison methodologies popular in the literature. We assume we are comparing two optimizers, or the same optimizer with different tuning parameters. For each optimizer, we have a number of approximation sets and we wish to determine which optimizer is the best.

10.4.2.1 Quality Indicators The most common approach consists of applying indicator functions on the approximation sets. These indicator functions or quality indicators convert each approximation set into a real number that is representative of the quality of an approximation set with respect to a specific property. Many indicators have been proposed, mainly to allow direct assessment of how good a solution is with respect to the objectives of MOO outlined in Section 10.2.3. Examples are

- *Cardinality* (i.e., number of solutions in the approximation set).
- *Generational distance*, which is a measure of the distance of an approximation set to the Pareto front (if known). It is the average distance between points on solution set and nearest point of the true Pareto front.
- *Spacing*, which quantifies the spread of solutions. It is the standard deviation of distance of points on solution front to nearest point on solution front.
- The *hypervolume indicator* measures the size of the dominated region bounded by some reference point [11].

The main attraction of these indicators is that statistical testing can be applied on the numbers they generate. However, they must be interpreted with care. Because they are measuring different aspects, it is frequently the case that two different indicators disagree on the superiority of different algorithms.

Some indicators can, even in nonpathological cases, judge an approximation set to be better than another when the latter dominates the former. The notion of indicator unreliability was introduced to formalize this property [25]. Formally, given two approximation sets A and B such that A dominates B, an unreliable indicator is an indicator that yields preference for the set B over A. For example, the cardinality indicator is unreliable. It can be shown that the spacing and generational distance indicators are unreliable as well.

Conversely, a reliable indicator *cannot* yield a preference for an approximation set D over another approximation set C when C dominates D. The hypervolume indicator is an example of a reliable indicator.

In addition to these unary indicators, binary quality indicators have been suggested. Their aim is to compare two approximation sets and give a quantitative measure of how much better set A is compared with set B. Binary indicator functions were shown to be more powerful than unary indicators.

10.4.2.2 Attainment Function Method This approach makes use of attainment surfaces. These are summary plots that delineate the region of the objective space that is dominated by the approximation set. It is possible to infer, from multiple runs of the optimizer, what is the probability that an objective vector is attained (i.e., weakly dominated by) an approximation set. The attainment function gives the probability that an objective vector is attained in one optimization run of the algorithm. By following this procedure for approximation sets generated from a large number of runs, one obtains an empirical attainment function, an estimate of the true attainment function. Statistical testing procedures can then be applied [25, 26] (Fig. 10.9).

The attainment function approach preserves more information than the quality indicators and the dominance ranking method. They are ideal for visual comparison and more sensitive to qualitative difference between optimizers than quality indicators. Their main drawback is that they are computationally expensive to generate and only practical for problems with two or three objective functions [27].

10.4.2.3 Dominance Ranking The dominance ranking approach was recently suggested in Ref. [25] as a preference-independent comparison method in an initial step to comparing MO optimizers. In this method, all approximation sets generated by the different optimizers are pooled, and then each approximation set is assigned a rank based on a ranking procedure. One such procedure could simply be the number of sets by which a specific approximation set is dominated. Once a rank is assigned to each set, the sets associated with each algorithm are transformed into a

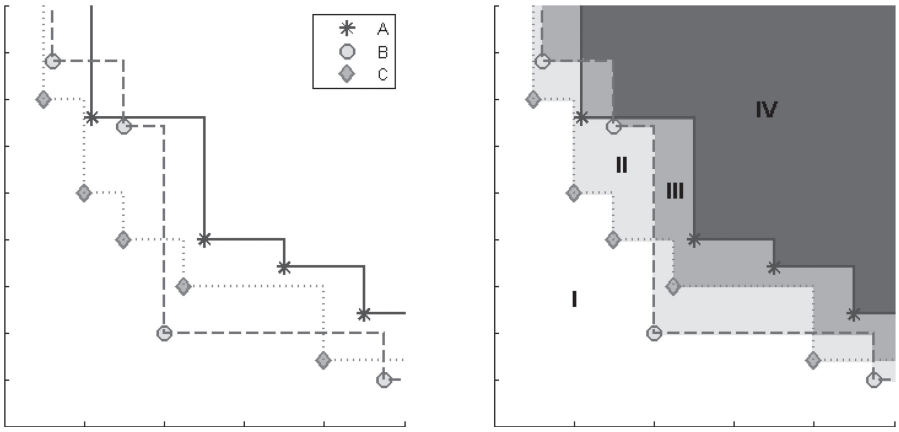

FIGURE 10.9 Attainment surfaces and empirical attainment functions. Left panel: Objective space representation of three approximation sets (A, B, and C), which are hypothetical outputs of a given MO optimizer for a biobjective minimization problem. The lines linking the objective vectors of each approximation set are the attainment surfaces. Right panel: Empirical attainment functions created from the three approximation sets, giving the probability that an objective vector is attained by the optimizer. In this case, objective vectors in regions I, II, III, and IV are respectively attained with probabilities 0, $1/3$, $2/3$, and 1.

set of values on which a statistical rank test can be applied to determine whether there is a significant statistical difference between the sets of ranks and by extension whether one algorithm outperforms another. If it is determined that an algorithm is significantly better than another, quality indicators and empirical attainment functions may be used to further characterize the differences in the optimizers, but are not necessary.

10.5 CONCLUSIONS

This chapter discussed to how to adapt population-based meta-heuristic algorithms, particularly EAs, to solving MO problems. As discussed, these problems are common in real-world optimization, especially in power systems. Thanks to the availability of cheap computational power, population-based techniques, which apply Pareto-optimality to concurrently optimize the different objectives, are practical, flexible turnkey approaches to generating approximation of Pareto fronts. They overcome the limitations of the classical methods such as aggregation-based techniques, which require *a priori* information about relative importance of the objectives and are sensitive to the shape of the Pareto front.

The main difficulty in MOO is that the solution consists of a set of trade-offs. As discussed in the chapter, Pareto-dominance can be incorporated in the basic operations of EAs (i.e., fitness evaluation, diversity, and elitism) to facilitate exploration and discovery of the Pareto front. The most popular algorithms in the literature to date were taken as illustrations. The subject of performance assessment, which is itself complicated by the need to compare sets of trade-offs, was briefly covered.

The hope is that this high-level overview of meta-heuristic MOO field has provided the reader with the basic tools to tackle MO problems. These algorithms are generic and flexible enough to be augmented and combined with application-specific knowledge to improve algorithmic performance and quality of the results.

ACKNOWLEDGMENTS

The authors would like to thank members of the University of Washington's Computational Intelligence Laboratory, especially Nissrine Krami and David Krout, for their input. Patrick Ngatchou was supported by the U.S. Office of Naval Research, contract no. N00014-01-G-0460.

REFERENCES

1. Stadler W. Multicriteria optimization in engineering and in the sciences. New York: Plenum Press; 1988.
2. Tabucanon MT. Multiple criteria decision making industry. Amsterdam: Elsevier Science; 1988.

3. Coello Coello CA, Van Veldhuizen DA, Lamont GB. Evolutionary algorithms for solving multi-objective problems. New York: Kluwer Academic Publishers; 2002.
4. Bagchi TP. Multiobjective scheduling by genetic algorithms. Boston: Kluwer Academic Publishers; 1999.
5. Abido MA. Environmental/economic power dispatch using multiobjective evolutionary algorithms: a comparative study. In: 2003 IEEE Power Engineering Society General Meeting, 13–17 July 2003, Toronto, Canada. IEEE: 2003.
6. Begovic M, Radibratovic B, Lambert F. On multiple Volt-VAR optimization in power systems. Proceedings of the 37th Hawaii International Conference on System Sciences, 2004.
7. Krami N, El-Sharkawi MA, Akherraz M. Multi-objective particle swarm optimization for reactive power planning. Accepted at the IEEE Swarm Intelligence Symposium, Indianapolis, 2006.
8. Jin Y, Olhofer M, Sendhoff B. Dynamic weighted aggregation for evolutionary multi-objective optimization: why does it work and how? Proc. Genetic and Evolutionary Computation Conf. (GECCO), 2001.
9. Parsopoulos KE, Vrahatis MN. Particle swarm optimization method in multiobjective problems. In: Proc. of the 2002 ACM Symposium on Applied Computing (SAC '02), Madrid, Spain. ACM Press: New York; 2002. p. 603–607.
10. Coello CAC. A comprehensive survey of evolutionary-based multiobjective optimization techniques. Knowledge and Information Systems 1999; 1(3):269–308.
11. Zitzler E, Deb K, Thiele L. Comparison of multiobjective evolutionary algorithms: empirical results. Evolutionary Computation 2000; 8(2):149–172.
12. Jones DF, Mirrazavi SK, Tamiz M. Multi-objective meta-heuristics: an overview of the current state-of-the-art. Eur J Operational Res 2002; 137(1):1–9.
13. Eiben AE, Smith JE. Introduction to evolutionary computing. Berlin: Springer; 2003.
14. Schaffer JD. Multiple objective optimization with vector evaluated genetic algorithms. In: Proceedings of the International Conference on Genetic Algorithms and Their Applications, Pittsburgh, IEEE; July 24–26, 1985. p. 93–100.
15. Fonseca C, Fleming PJ. Genetic algorithms for multiobjective optimization: formulation, discussion, generalization. In: Proceedings of the Fifth International Conference on Genetic Algorithms, San Mateo, CA, IEEE; 1993. p. 416–423.
16. Zitzler E, Thiele L. An evolutionary algorithm for multiobjective optimization: the strength Pareto approach. TIK Tech. Report No. 43, Swiss Federal Institute of Technology (ETH); 1998.
17. Srinivas N, Deb K. Multiobjective function optimization using nondominated sorting genetic algorithms. Evolutionary Computation 1994; 2(3):221–248.
18. Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA–II. IEEE Trans Evol Computat 2002; 6(2):182–197.
19. Horn J, Nafpliotis N, Goldberg DE. A niched Pareto genetic algorithm for multiobjective optimization. In: Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, Piscataway, NJ. Vol. 1. IEEE; 1994, p. 82–87.
20. Zitzler E, Laumanns M, Thiele L. SPEA2: improving the strength Pareto evolutionary algorithm. TIK Report No. 103, ETH Zurich; 2001.

21. Knowles J, Corne D. The Pareto archived evolution strategy: a new baseline algorithm for Pareto multiobjective optimisation. In: Proceedings of the 1999 Congress on Evolutionary Computation, IEEE; 1999, Vol. 1, p. 98–105.
22. Kennedy J, Eberhart RC. Swarm intelligence. Morgan Kaufmann; 2001.
23. Song M-P, Gu G-C. Research on particle swarm optimization: a review. In: Proceedings of the Third International Conference on Machine Learning and Cybernetics. IEEE; 2004; p. 2216–2241.
24. Coello Coello CA, Pulido GT, Lechuga MS. Handling multiple objectives with particle swarm optimization. IEEE Trans Evol Computat 2004; 8(3):256–279.
25. Knowles JD, Thiele L, Zitzler E. A tutorial on the performance assessment of stochastic multiobjective optimizers. TIK Report No. 214, ETH Zurich; July 2005.
26. da Fonseca VG, Fonseca CM, Hall AO. Inferential performance assessment of stochastic optimisers and the attainment function. Lecture Notes in Computer Science, Volume 1993, IEEE; Jan 2001, p. 213–225.
27. Knowles JD. A summary-attainment-surface plotting method for visualizing the performance of stochastic multiobjective optimizers. Proceedings 5th International Conference on Intelligent Systems Design and Applications, IEEE; 2005. p. 552–557.