Multi-Objective Optimization Using Genetic Algorithms: A Tutorial

Abdullah Konak[1], David W. Coit[2], Alice E. Smith[3]

[1]Information Sciences and Technology, Penn State Berks-Lehigh Valley

[2]Department of Industrial and Systems Engineering, Rutgers University

[3]Department of Industrial and Systems Engineering, Auburn University

abstract – Multi-objective formulations are a realistic models for many complex engineering optimization problems. Customized genetic algorithms have been demonstrated to be particularly effective to determine excellent solutions to these problems. In many real-life problems, objectives under consideration conflict with each other, and optimizing a particular solution with respect to a single objective can result in unacceptable results with respect to the other objectives. A reasonable solution to a multi-objective problem is to investigate a set of solutions, each of which satisfies the objectives at an acceptable level without being dominated by any other solution. In this paper, an overview and tutorial is presented describing genetic algorithms developed specifically for these problems with multiple objectives. They differ from traditional genetic algorithms by using specialized fitness functions, introducing methods to promote solution diversity, and other approaches.

1. Introduction

The objective of this paper is present an overview and tutorial of multiple-objective optimization methods using genetic algorithms (GA). For multiple-objective problems, the objectives are generally conflicting, preventing simultaneous optimization of each objective. Many, or even most, real engineering problems actually do have multiple-objectives, i.e., minimize cost, maximize performance, maximize reliability, etc. These are difficult but realistic problems. GA are a popular meta-heuristic that is particularly well-suited for this class of problems. Traditional GA are customized to accommodate multi-objective problems by using specialized fitness functions, introducing methods to promote solution diversity, and other approaches.

There are two general approaches to multiple-objective optimization. One is to combine the individual objective functions into a single composite function. Determination of a single objective is possible with methods such as utility theory, weighted sum method, etc., but the problem lies in the correct selection of the weights or utility functions to characterize the decision-makers preferences. In practice, it can be very difficult to precisely and accurately select these weights, even for someone very familiar with the problem domain. Unfortunately, small perturbations in the weights can lead to very different solutions. For this reason and others, decision-makers often prefer a set of promising solutions given the multiple objectives.

The second general approach is to determine an entire Pareto optimal solution set or a representative subset. A Pareto optimal set is a set of solutions that are nondominated with respect to each other. While moving from one Pareto solution to another, there is always a certain amount of sacrifice in one objective to achieve a certain amount of gain in the other. Pareto optimal solution sets are often preferred to single solutions because they can be practical when considering real-life problems, since the final solution of the decision maker is always a trade-off between crucial parameters. Pareto optimal sets can be of varied sizes, but the size of the Pareto set increases with the increase in the number of objectives.

2. Multi-Objective Optimization Formulation

A multi-objective decision problem is defined as follows: Given an n-dimensional decision variable vector $\mathbf{x}=\{x_1,\ldots,x_n\}$ in the solution space $\mathbf{X}$, find a vector $\mathbf{x}^*$ that minimizes a given set of K objective functions $\mathbf{z}(\mathbf{x}^*)=\{z_1(\mathbf{x}^*),\ldots,z_K(\mathbf{x}^*)\}$. The solution space $\mathbf{X}$ is generally restricted by a series of constraints, such as $g_j(\mathbf{x}^*)=b_j$ for $j = 1, \ldots, m$, and bounds on the decision variables.

In many real-life problems, objectives under consideration conflict with each other. Hence, optimizing $\mathbf{x}$ with respect to a single objective often results in unacceptable results with respect to the other objectives. Therefore, a perfect multi-objective solution that simultaneously optimizes each objective function is almost impossible. A reasonable solution to a multi-objective problem is to investigate a set of solutions, each of which satisfies the objectives at an acceptable level without being dominated by any other solution.

If all objective functions are for minimization, a feasible solution $\mathbf{x}$ is said to dominate another feasible solution $\mathbf{y}$ ($\mathbf{x} \succ \mathbf{y}$), if and only if, $z_i(\mathbf{x}) \le z_i(\mathbf{y})$ for $i=1, \ldots, K$ and $z_j(\mathbf{x}) < z_j(\mathbf{y})$ for least one objective function j. A solution is said to be *Pareto optimal* if it is not dominated by any other solution in the solution space. A Pareto optimal solution cannot be improved with respect to any objective without worsening at least one other objective. The set of all feasible non-dominated solutions in **X** is referred to as the *Pareto optimal set,* and for a given Pareto optimal set, the corresponding objective function values in the objective space is called the *Pareto front*. For many problems, the number of Pareto optimal solutions is enormous (maybe infinite).

The ultimate goal of a multi-objective optimization algorithm is to identify solutions in the Pareto optimal set. However, identifying the entire Pareto optimal set, for many multi-objective problems, is practically impossible due to its size. In addition, for many problems, especially for combinatorial optimization problems, proof of solution optimality is computationally infeasible. Therefore, a practical approach to multi-objective optimization is to investigate a set of solutions (*the best-known Pareto set*) that represent the Pareto optimal set as much as possible. With these concerns in mind, a multi-objective optimization approach should achieve the following three conflicting goals:

1. The best-known Pareto front should be as close possible as to the true Pareto front. Ideally, the best-known Pareto set should be a subset of the Pareto optimal set.
2. Solutions in the best-known Pareto set should be uniformly distributed and diverse over of the Pareto front in order to provide the decision maker a true picture of trade-offs.
3. In addition, the best-known Pareto front should capture the whole spectrum of the Pareto front. This requires investigating solutions at the extreme ends of the objective function space.

This paper presents common approaches used in multi-objective genetic algorithms to attain these three conflicting goals while solving a multi-objective optimization problem.

3. Genetic Algorithms

The concept of genetic algorithms (GA) was developed by Holland and his colleagues in the 1960s and 1970s [18]. GA is inspired by the evolutionist theory explaining the origin of species. In nature, weak and unfit species within their environment are faced with extinction by natural selection. The strong ones have greater opportunity to pass their genes to future generations via reproduction. In the long run, species carrying the correct combination in their genes become dominant in their population. Sometimes, during the slow process of evolution, random changes may occur in genes. If these changes provide additional advantages in the challenge for survival, new species evolve from the old ones. Unsuccessful changes are eliminated by natural selection.

In GA terminology, a solution vector $\mathbf{x} \in \mathbf{X}$ is called an individual or a *chromosome*. Chromosomes are made of discrete units called *genes*. Each gene controls one or more features of the chromosome. In the original implementation of GA by Holland, genes are assumed to be binary numbers. In later implementations, more varied gene types have been introduced. Normally, a chromosome corresponds to a unique solution $\mathbf{x}$ in the solution space. This requires a mapping mechanism between the solution space and the chromosomes. This mapping is called an encoding. In fact, GA works on the *encoding* of a problem, not on the problem itself.

GA operates with a collection of chromosomes, called a *population*. The population is normally randomly initialized. As the search evolves, the population includes fitter and fitter solutions, and eventually it converges, meaning that it is dominated by a single solution. Holland also presented a proof of convergence (the schema theorem) to the global optimum where chromosomes are binary vectors.

GA use two operators to generate new solutions from existing ones: *crossover* and *mutation*. The crossover operator is the most important operator of GA. In crossover, generally two chromosomes, called *parents*, are combined together to form new chromosomes, called *offspring*. The parents are selected among existing chromosomes in the population with preference towards fitness so that offspring is expected to inherit good genes which make the parents fitter. By iteratively applying the crossover operator, genes of good chromosomes are expected to appear more frequently in the population, eventually leading to convergence to an overall good solution.

The mutation operator introduces random changes into characteristics of chromosomes. Mutation is generally applied at the gene level. In typical GA implementations, the mutation rate (probability of changing the properties of a gene) is very small, typically less than 1%. Therefore, the new chromosome produced by mutation will not be very different from the original one. Mutation plays a critical role in GA. As discussed earlier, crossover leads the population to converge by making the chromosomes in the population alike. Mutation reintroduces genetic diversity back into the population and assists the search escape from local optima.

Reproduction involves selection of chromosomes for the next generation. In the most general case, the fitness of an individual determines the probability of its survival for the next generation. There are different selection procedures in GA depending on how the fitness values are used. Proportional selection, ranking, and tournament selection are the most popular selection procedures. The procedure of a generic GA is given as follows:

Step 1. Set t =1. Randomly generate N solutions to form the first population, P_1. Evaluate the fitness of solutions in P_1.

Step 2. **Crossover**: Generate an offspring population Q_t as follows.

2.1. Choose two solutions $\mathbf{x}$ and $\mathbf{y}$ from P_t based on the fitness values.

2.2. Using a crossover operator, generate offspring and add them to Q_t.

Step 3. **Mutation:** Mutate each solution $\mathbf{x} \in Q_t$ with a predefined mutation rate.

Step 4. **Fitness Assignment:** Evaluate and assign a fitness value to each solution $\mathbf{x} \in Q_t$ based its objective function value and infeasibility.

Step 5. **Selection:** Select N solutions from Q_t based on their fitness and assigned them P_{t+1}.

Step 6. If the stopping criterion is satisfied, terminate the search and return the current population, else, set t=t+1 go to Step 2.

4. Multi-objective Genetic Algorithms

Being a population based approach, GA are well suited to solve multi-objective optimization problems. A generic single-objective GA can be easily modified to find a set of multiple non-dominated solutions in a single run. The ability of GA to simultaneously search different regions of a solution space makes it possible to find a diverse set of solutions for difficult problems with non-convex, discontinuous, and multi-modal solutions spaces. The crossover operator of GA may exploit structures of good solutions with respect to different objectives to create new non-dominated solutions in unexplored parts of the Pareto front. In addition, most multi-objective GA do not require the user to prioritize, scale, or weigh objectives. Therefore, GA has been the most popular heuristic approach to multi-objective design and optimization problems. Jones et al. [25] reported that 90% of the approaches to multi-objective optimization aimed to approximate the true Pareto front for the underlying problem. A majority of these used a meta-heuristic technique, and 70% of all meta-heuristics approaches were based on evolutionary approaches.

The first multi-objective GA, called Vector Evaluated Genetic Algorithms (or VEGA), was proposed by Schaffer [44]. Afterward, several major multi-objective evolutionary algorithms were developed such as Multi-objective Genetic Algorithm (MOGA) [13], Niched Pareto

Genetic Algorithm [19], Random Weighted Genetic Algorithm (RWGA)[39], Nondominated Sorting Genetic Algorithm (NSGA) [45], Strength Pareto Evolutionary Algorithm (SPEA) [55], Pareto-Archived Evolution Strategy (PAES) [27], Fast Non-dominated Sorting Genetic Algorithm (NSGA-II) [9], Multi-objective Evolutionary Algorithm (MEA) [42], Rank-Density Based Genetic Algorithm (RDGA) [32]. Note that although there are many variations of multi-objective GA in the literature, these cited GA are well-known and credible algorithms that have been used in many applications and their performances were tested in several comparative studies.

Several survey papers [1-3, 12, 14, 22, 51, 54, 55] have been published on evolutionary multi-objective optimization. Coello Coello lists more than 1800 references in his website [4]. Most survey papers on multi-objective evolutionary approaches introduce and compare different algorithms. This paper takes a different course and focuses on important issues while designing a multi-objective GA and describes common techniques used in multi-objective GA to attain the three goals in multi-objective optimization.

4.1. Fitness Functions

4.1.1. Weighted Sum Approaches.

The classical approach to solve a multi-objective optimization problem is to assign a weight w_i to each normalized objective function $z_i'(\mathbf{x})$ so that the problem is converted to a single objective problem with a scalar objective function as follows:

$$\min z = w_1 z_1'(\mathbf{x}) + w_2 z_2'(\mathbf{x}) + \ldots + w_k z_k'(\mathbf{x}) \tag{1}$$

where $z_i'(\mathbf{x})$ is the normalized objective function, $z_i(\mathbf{x})$ and $\sum w_i = 1$. This approach is *called a priori* approach since the user is expected to provide the weights. Solving a problem with the objective function (1) for a given weight vector $\mathbf{w} = \{w_1, w_2, \ldots, w_k\}$ yields a single solution, and if multiple solutions are desired, the problem should be solved multiple times with different weight combinations. The main difficulty with this approach is selecting a weight vector for each run. To automate this process, Hajela and Lin [17] proposed the weight-based genetic algorithm for multi-objective optimization (WBGA-MO). In the WBGA-MO, each solution $\mathbf{x_i}$ in the population uses a different weight vector $\mathbf{w_i} = \{w_1, w_2, \ldots, w_k\}$ in the calculation of objective function (1). The weight vector $\mathbf{w_i}$ is embedded within the chromosome of solution $\mathbf{x_i}$.

Therefore, multiple solutions can be simultaneously searched in a single run. In addition, weight vectors can be adjusted to promote diversity of the population.

Other researchers [39, 40] have proposed a multi-objective genetic algorithm based on a weighted sum of multiple objective functions where a normalized weight vector $\mathbf{w_i}$ is randomly generated for each solution $\mathbf{x_i}$ during the selection phase at each generation. This approach aims to stipulate multiple search directions in a single run without using any additional parameters.

The main advantage of the weighted sum approach is a straightforward implementation. Since a single objective is used in fitness assignment, a single objective GA can be used with minimum modifications. In addition, this approach is computationally very efficient. The main disadvantage of this approach is that not all Pareto-optimal solutions can be investigated when the true Pareto front is non-convex. Therefore, the multi-objective genetic algorithms based on the weighed sum approach have difficulty in finding solutions uniformly distributed over a non-convex trade-off surface [54].

4.1.2. Altering Objective Functions.

As mentioned earlier, the VEGA [44] is the first GA used to approximate the Pareto optimal set by a set of non-dominated solutions. In the VEGA, population P_t is randomly divided into K equal sized sub-populations; P_1, P_2, ..., P_K. Then, each solution in subpopulation P_i is assigned a fitness value based on objective function z_i. Solutions are selected from these subpopulations using proportional selection for crossover and mutation. Crossover and mutation are performed on the new population in the same way with the single objective GA. A similar approach is to use only a single objective function which is randomly determined each time in the selection phase [31].

These approaches are easy to implement and computationally as efficient as a single-objective GA. The major drawback of objective switching is that the population tends to converge to solutions which are very superior in one objective, but very poor at others.

4.1.3. Pareto-Ranking Approaches.

Pareto-ranking approaches explicitly utilize the concept of Pareto dominance in evaluating fitness or assigning selection probability to solutions. The population is ranked according to a dominance rule, and then each solution is assigned a fitness value based on its rank in the population, not its actual objective function value. Note that herein all objectives are assumed to be minimized. Therefore, a lower rank corresponds to a better solution in the following discussions.

The first Pareto ranking technique was proposed by Goldberg [15] as follows:

Step 1. Set i=1 and TP=P

Step 2. Identify non-dominated solutions in TP and assigned them set to F_i.

Step 3. Set $TP = TP \setminus F_i$. If TP=∅ go to Step 4, else set i=i+1 and go to Step 2.

Step 4. For every solution $\mathbf{x} \in P$ at generation t, assign rank $r_1(\mathbf{x},t) = i$ if $\mathbf{x} \in F_i$.

In the procedure above, F_1, F_2, ... are called non-dominated fronts, and F_1 is the Pareto front of population P. Fonseca and Fleming [13] used a slightly different rank assignment approach follows:

$$r_2(\mathbf{x},t) = 1 + nq(\mathbf{x},t)$$

where $nq(\mathbf{x},t)$ is the number of solutions dominating solution $\mathbf{x}$ at generation t. This ranking method penalizes solutions located in the regions of the objective function space which are dominated (covered) by densely populated sections of the Pareto front. For example, in Figure 1b solution **i** is dominated by solutions **c, d** and **e.** Therefore, it is assigned a rank of 4 although it is in the same front with solutions **f, g** and **h** which are dominated by only a single solution.

The SPEA [55] uses a ranking procedure to assign better fitness values to non-dominated solutions at underrepresented regions of the objective space. In the SPEA, an external list E of a fixed size stores non-dominated solutions that have been investigated thus far during the search. For each solution $\mathbf{y} \in E$, a strength value is defined as,

$$s(\mathbf{y},t) = \frac{np(\mathbf{y},t)}{N_P + 1}$$

where $np(\mathbf{y},t)$ is the number solutions that $\mathbf{y}$ dominates in P. The rank $r(\mathbf{y},t)$ of a solution $\mathbf{y} \in E$ is assigned as $r_3(\mathbf{y},t) = s(\mathbf{y},t)$ and the rank of a solution $\mathbf{x} \in P$ is calculated as,

$$r_3(\mathbf{x},t) = 1 + \sum_{\mathbf{y} \in E, \mathbf{y} \succ \mathbf{x}} s(\mathbf{y},t)$$

Figure 1c illustrates an example of the SPEA ranking method. In the former two methods, all non-dominated solutions are assigned a rank of 1. This method, however, favors solution **a** (in the figure) over the other non-dominated solutions since it covers the least number of solutions in the objective function space. Therefore, a wide, uniformly distributed set of non-dominated solutions is encouraged.

Accumulated ranking density strategy [32] also aims to penalize redundancy in the population due to overrepresentation. This ranking method is given as,

$$r_4(\mathbf{x},t) = 1 + \sum_{\mathbf{y}\in P, \mathbf{y}\succ\mathbf{x}} r(\mathbf{y},t)$$

To calculate the rank of a solution **x**, the rank of the solutions dominating this solution must be calculated first. Figure 1d shows an example of this ranking method (based on r_2). Using ranking method r_4, solutions **i**, **l** and **n** are ranked higher than their counterparts at the same non-dominated front since the portion of the trade-off surface covering them is crowded by three nearby solutions **c, d** and **e**.

4.2. Diversity: Fitness Assignment, Fitness Sharing, and Niching.

Maintaining a diverse population is an important consideration in multi-objective GA to obtain solutions uniformly distributed over the true Pareto font. Without taking any preventive measures, the population tends to form relatively few clusters in multi-objective GA. This phenomenon is called genetic drift, and several approaches are used to prevent genetic drift, as follows.

4.2.1. Fitness Sharing

Fitness sharing aims to encourage the search in unexplored sections of a Pareto front by artificially reducing fitness of solutions in densely populated areas. To achieve this goal, densely populated areas are identified and a fair penalty method is used to penalize the solutions located in such areas.

The idea of fitness sharing was first proposed by Goldberg and Richardson [16] in the investigation of multiple local optima for multi-modal functions. Fonseca and Fleming [13] used this idea to penalize clustered solutions with the same rank as follows.

Step 1. Calculate the Euclidean distance between every solution pair **x** and **y** in the normalized objective space between 0 and 1 as

$$dz(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{k=1}^{K}\left(\frac{z_k(\mathbf{x}) - z_k(\mathbf{y})}{z_k^{\max} - z_k^{\min}}\right)^2} \qquad (2)$$

where $z_k^{\max}$ and $z_k^{\min}$ are the maximum and minimum value of the objective function $z_k(\cdot)$ observed so far during the search, respectively.

Step 2. Based on these distances, calculate a niche count for each solution $\mathbf{x}\in P$ as

$$nc(\mathbf{x},t) = \sum_{\substack{\mathbf{y}\in P \\ r(\mathbf{y},t)=r(\mathbf{x},t)}} \max\left\{\frac{\sigma_{\text{share}} - d(\mathbf{x},\mathbf{y})}{\sigma_{\text{share}}}, 0\right\}$$

where σ_{share} is the niche size.

Step 3. After calculating niche counts, the fitness of each solution is adjusted as follows:

$$f'(\mathbf{x},t) = \frac{f(\mathbf{x},t)}{nc(\mathbf{x},t)}$$

In the procedure above, σ_{share} defines a neighborhood of solutions in the objective space (Figure 1a). The solutions in the same neighborhood contribute to each other's niche count. Therefore, a solution in a crowded neighborhood will have a higher niche count reducing the probability of selecting that solution as a parent. As a result, niching limits the proliferation of solutions in one particular neighborhood of the objective function space.

Another alternative is to use the Hamming distance (the distance in the decision variable space) between two solutions **x** and **y** which is defined as

$$dx(\mathbf{x},\mathbf{y}) = \sqrt{\frac{1}{M}\sum_{i=1}^{M}(x_i - y_i)^2} \tag{3}$$

in the calculation of niche count. Equation (3) is a measure of structural differences between two solutions. Two solutions might be very close in the objective function space while they have very different structural features. Therefore, fitness sharing based on the objective function space may reduce diversity in the decision variable space. However, Deb and Goldberg [8] reported that fitness sharing on the objective function space usually performs better than one based on the decision variable space.

One of the disadvantages of the fitness sharing based on niche count is that the user has to select a new parameter σ_{share}. To address this problem, Deb and Goldberg [8] and Fonseca and Fleming [13] developed systematic approaches to estimate and dynamically update σ_{share}. Another disadvantage of niching is computational effort to calculate niche counts. However, benefits of fitness sharing surpass the burden of extra computational effort in many applications. Miller and Shaw [36] proposed a dynamic niche sharing approach to increase effectiveness of computing niche counts.

4.2.2. Crowding Distance

Crowding distance approaches aim to obtain a uniform spread of solutions along the best-known Pareto front without using a fitness sharing parameter. For example, the NSGA-II [9] use a crowding distance method as follows (Figure 2b):

Step 1. Rank the population and identify non-dominated fronts F_1, F_2, ..., F_R. For each front j=1, ..., R repeat Steps 2 and 3.

Step 2. For each objective function k, sort the solutions in F_j in the ascending order. Let l=|F_j| and $\mathbf{x}_{[i,k]}$ represent the i^{th} solution in the sorted list with respect to the objective function k. Assign $cd_k(\mathbf{x}_{[1,k]}) = \infty$ and $cd_k(\mathbf{x}_{[l,k]}) = \infty$, and for i=2, ..., l assign

$$cd_k(\mathbf{x}_{[i,k]}) = \frac{z_k(\mathbf{x}_{[i+1,k]}) - z_k(\mathbf{x}^{k}_{[i-1,k]})}{z_k^{\max} - z_k^{\min}}$$

Step 3. To find the total crowding distance $cd(\mathbf{x})$ of a solution $\mathbf{x}$, sum the solution crowding distances with respect to each objective, i.e., $cd(\mathbf{x}) = \sum_k cd_k(\mathbf{x})$.

The main advantage of the crowding approach described above is that a measure of population density around a solution is computed without requiring a user-defined parameter. In the NSGA-II, this crowding distance measure is used as a tie-breaker as in the selection phase that follows. Randomly select two solutions $\mathbf{x}$ and $\mathbf{y}$; if the solutions are in the same non-dominated front, the solution with a higher crowding distance wins. Otherwise, the solution with the lowest rank is selected.

4.2.3. Cell-Based Density

In this approach [26, 27, 32, 53], the objective space is divided into K-dimensional cells (see Figure 2c). The number of solutions in each cell is defined as the density of the cell, and the density of a solution is equal to the density of the cell in which the solution is located. This density information is used to achieve diversity similarly to the fitness sharing approach. For example, in the PAES [26, 27], between two non-dominated solutions, the one with a lower density is preferable.

Lu and Yen [32, 53] developed an efficient approach to identify a solution's cell in case of dynamic cell dimensions. In this approach, the width of a cell along the kth objective dimension is $(z_k^{\max} - z_k^{\min}) / n_k$ where n_k is the number cells dedicated the kth objective dimension and $z_k^{\max}$ and $z_k^{\min}$ are the maximum and minimum values of the objective function k so far in the search, respectively. Therefore, cell boundaries are updated when a new maximum or minimum objective function value is discovered.

The main advantage of the cell based density approach is that a global density map of the objective function space is obtained as a result of the density calculation. The search can be encouraged toward sparsely inhabited regions of the objective function space based on this map. The RDGA [32] uses a method based on this global density map to push solutions out of high density areas to low density areas.

4.3. Elitisim

Elitism in the context of single-objective GA means that the best solution found so far during the search has immunity against selection and always survives in the next generation. In this respect, all non-dominated solutions discovered by a multi-objective GA are considered as elite solutions. However, implementation of elitism in multi-objective optimization is not as straightforward as in single objective optimization mainly due to the large number of possible elitist solutions. Earlier multi-objective GA did not use elitism. However, most recent multi-objective GA and their variations use elitism. As discussed in [6, 47, 55], multi-objective GA using elitist strategies tend to outperform their non-elitist counterparts. Multi-objective GA use two strategies to implement elitism [22]: (i) maintaining elitist solutions in the population, and (ii) storing elitist solutions in an external secondary list and reintroducing them to the population.

4.3.1. Strategies to Maintain Elitist Solutions in the Population

Random selection does not ensure that a non-dominated solution will survive in the next generation. A straightforward implementation of elitism in a multi-objective GA is to copy all non-dominated solution in population P_t to population P_{t+1}, then fill the rest of P_{t+1} by selecting from the remaining dominated solutions in P_t. This approach will not work when the total number of non-dominated parent and offspring solutions is larger than N_P. To address this problem, several approaches have been proposed.

Konak and Smith [29, 30] proposed a multi-objective GA with dynamic population size and a pure elitist strategy. In this multi-objective GA, the population includes only non-dominated solutions. If the size of the population reaches an upper bound $N_{\max}$, $N_{\max}$-$N_{\min}$ solutions are removed from the population giving consideration to maintaining the diversity of the current non-dominated front. To achieve this, the Pareto domination tournament selection is used as follows [19]. Two solutions are randomly chosen and the solution with the higher niche count is removed since all solutions are non-dominated. A similar pure elitist multi-objective GA with a dynamic population size has also been proposed [42].

The NSGA-II uses a fixed population size of N. In generation t, an offspring population Q_t of size N is created from parent population P_t and non-dominated fronts F_1, F_2, ..., F_R are identified in the combined population $P_t \cup Q_t$. The next population P_{t+1} is filled starting from solutions in F_1, then F_2, and so on as follows. Let k be the index of a non-dominated front F_k that $|F_1 \cup F_2 \cup ... \cup F_k| \leq N$ and $|F_1 \cup F_2 \cup ... \cup F_k \cup F_{k+1}| > N$. First, all solutions in fronts F_1, F_2, ..., F_k are copied to P_{t+1}, and then the least crowded (N-$|P_{t+1}|$) solutions in F_{k+1} are added to P_{t+1}. This approach makes sure that all non-dominated solutions (F_1) are included in the next population if $|F_1| \leq N$, and otherwise the selection based on a crowding distance will promote diversity.

4.3.2. Elitism with External Populations

When an external list is used to store elitist solutions, several issues must be addressed. The first issue is which solutions are going to be stored in elitist list E. Most multi-objective GA store non-dominated solutions investigated so far during the search [55], and E is updated each time a new solution is created by removing elitist solutions dominated by the new solution or adding the new solution if it is not dominated by any existing elitist solution. This is a computationally expensive operation. Several data structures were proposed to efficiently store, update, search in list E [11, 38]. Another issue is the size of list E. Since there might possibly exist a very large number of Pareto optimal solutions for a problem, the elitist list can grow extremely large. Therefore, pruning techniques were proposed to control the size of E. For example, the SPEA uses the average linkage clustering method [37] to reduce the size of E to an upper limit N when the number of the non-dominated solutions exceeds N as follows.

Step 1. Initially, assign each solution $\mathbf{x} \in E$ to a cluster c_i, $C = \{c_1, c_2, \ldots, c_M\}$

Step 2. Calculate the distance between all pairs of clusters c_i and c_j as follows

$$d_{c_i,c_j} = \frac{1}{|c_i| \cdot |c_j|} \sum_{\mathbf{x} \in c_i, \mathbf{y} \in c_j} d(\mathbf{x}, \mathbf{y})$$

Here, the distance $d(\mathbf{x}, \mathbf{y})$ can be calculated in the objective function space using equation (2) or in the decision variable space using equation (3).

Step 3. Merge the cluster pair c_i and c_j with the minimum distance among all clusters into a new cluster.

Step 4. If $|C| \leq N$, go to Step 5, else go to Step 2.

Step 5. For each cluster, determine a solution with the minimum average distance to all other solutions in the same cluster (called centroid solution). Keep the centroid solutions for every cluster and remove other solutions from E.

The final issue is the selection of elitist solutions from E to be reintroduced to the population. In [32, 53, 55], solutions for P_{t+1} are selected from the combined population of P_t and E_t. To implement this strategy, population P_t and E_t are combined together, a fitness value is assigned to each solution in the combined population $P_t \cup E_t$, and then, N solutions are selected for the next generation P_{t+1} based on the assigned fitness values. Another strategy is to reserve a room for n elitist solutions in the next population [20]. In this strategy, N - n solutions are selected from parents and newly created offspring and n solutions are selected from E_t.

4.4. Constraint Handling

Most real-world optimization problems include constraints that must be satisfied. Single-objective GA use four different constraint handling strategy: (i) discarding infeasible solutions, (ii) reducing the fitness of infeasible solutions by using a penalty function, (iii) if possible, customizing genetic operators to always produce feasible solutions, and (iv) repairing infeasible solutions. Handling of constraints has not been adequately researched for multi-objective GA [23]. For instance, all major multi-objective GA assumed problems without any constraints. While constraint handling strategies (i), (iii), and (iv) are directly applicable in the multi-objective case, implementation of penalty function strategies, which is by far the most frequently used constraint handling strategy in single-objective GA, is not straightforward in multi-objective GA, mainly due to fact that fitness assignment is usually based on the non-dominance rank of a solution, not on its objective function values.

Jimenez et al. [24] proposed a niched selection strategy to address infeasibility in multi-objective problems as follows:

Step 1. Randomly chose two solutions **x** and **y** from the population.

Step 2. If one of the solutions is feasible and the other one is infeasible, the winner is the feasible solution, and stop. Otherwise, if both solutions are infeasible go to Step 3, else go to step 4.

Step 3. In this case, solutions **x** and **y** are both infeasible. Then, select a random reference set C among infeasible solutions in the population. Compare solutions **x** and **y** to the solutions in reference set C with respect to their degree of infeasibility. In order to achieve this, calculate a measure of infeasibility (e.g., the number of constraints violated or total constraint violation) for solutions **x**, **y**, and in set *C*. If one of solutions **x** and **y** is better and the other one is worse than the best solution in *C*, with respect to the calculated infeasibility measure, then the winner is the least infeasible solution. However, if there is a tie, that is both solutions **x** and **y** are either better or worse than the best solution in *C*, then their niche counts in the decision variable space (equation (3)) is used for selection. In this case, the solution with the lower niche count is the winner.

Step 4. In this case, solutions **x** and **y** are both feasible. Then, select a random reference set *C* among feasible solutions in the population. Compare solutions **x** and **y** to the solutions in set *C*. If one of them is non-dominated in set *C*, and the other is dominated by at least one solution, the winner is the former. Otherwise, there is a tie between solutions **x** and **y**, and the niche count of the solutions are calculated in the decision variable space. The solution with the smaller niche count is the winner of the tournament selection.

The procedure above is a comprehensive approach to deal with infeasibility while maintaining diversity and dominance of the population. Main disadvantages of this procedure are its computational complexity and additional parameters such as the size of reference set *C* and niche size. Modifications are also possible. In Step 4, for example, the niche count of the solutions can be calculated in the objective function space instead of the decision variable space. In Step 3, the solution with the least infeasibility can be declared as the winner without comparing solutions **x** and **y** to a reference set *C* with respect to infeasibility. Such modifications can reduce the computational complexity of the procedure.

Deb [9] proposed the constrain-domination concept and a binary tournament selection method based on it, called a constrained tournament method. A solution **x** is said to constrain-dominate a solution **y** if either of the following cases are satisfied:

Case 1: Solution **x** is feasible and solution **y** is infeasible.

Case 2: Solutions **x** and **y** are both infeasible; however, solution **x** has a smaller constraint violation than **y.**

Case 3: Solutions **x** and **y** are both feasible, and solution **x** dominates solution **y**.

In the constraint tournament method, first non-constrain-dominance fronts F_1, F_2, F_3, ...,

F_R are identified in a similar way defined in [15], but by using the constrain-domination criterion instead of the regular domination concept. Note that set F_1 corresponds to the set of feasible non-dominated solutions in the population and front F_i is more preferred than F_j for $i<j$. In the constraint tournament selection, two solutions **x** and **y** are randomly chosen from the population. Between **x** and **y**, the winner is the one in a more preferred non-constrain-dominance front. If solutions **x** and **y** are both in the same front, then the winner is decided based on niche counts or crowding distances of the solution. The main advantages of the constrained tournament method are that it requires fewer parameters and it can be easily integrated to multi-objective GA.

4.5. Parallel and Hybrid Multi-Objective GA

All comparative studies on multi-objective GA agree that elitism and diversity preservation mechanisms improve performance of multi-objective GA. However, implementing elitism and diversity preservation strategies usually require substantial computational effort and computer memory. In addition, evaluation of objective functions may take considerable time in real-life problems. Therefore, researchers have been interested in reducing execution time and resource requirement of multi-objective GA using advanced data structures. One of the latest trends in this avenue is parallel and distributed processing. Several recent papers [5, 48-50] presented parallel implementation of multi-objective GA over multiple processors.

Hybridization of GA with local search algorithms is frequently applied in single-objective GAs. This approach is usually referred to as a memetic algorithm. Generally, a local search algorithm proceeds as follows.

Step 1. Start with an initial solutions **x.**

Step 2. Generate a set of neighbor solutions around solution **x** using a simple perturbation rule.

Step 3. If the best solution in the neighborhood set is better than **x**, replace **x** with this solution and go to Step 2, else stop.

A local search algorithm is particularly effective in finding local optima if the solution space around the initial solution is convex. This is usually difficult to achieve using standard GA operators. In hybridization of multi-objective GA with local search algorithms, important issues are: (i) selecting a solution to apply the local search and (ii) identifying a solution in the neighborhood as the new best solution when multiple non-dominated local solutions exist. Several approaches have been proposed to address these two issues as follows.

Paquete and Stutzle [41] described a bi-objective GA where a local search is used to generate initial solutions by optimizing only one single objective. Deb and Goel [7] applied a local search to only final solutions. In Ishibuchi and Murata's approach [20], a local search procedure is applied to each offspring generated by crossover, using the same weight vector of the offspring's parents to evaluate neighborhood solutions. Similarly, Ishibuchi [21] also used the weighted sum of the objective functions to evaluate solution during the local search. However, the local search is selectively applied to only promising solutions, and weights are also randomly generated, instead of using the parents' weight vector. Knowles and Corne [28] presented a memetic version of the PAES, called M-PAES. The PAES uses the dominance concept to evaluate solutions. Therefore, in M-PAES, a set of local non-dominated solutions is used as a comparison set for solutions investigated during the local search. When a new solution is created in the neighborhood, it is only compared with this local non-dominated set and necessary updates are performed. The local search is terminated after a maximum number of local solutions are investigated or a maximum number of local moves are performed without any improvement. Tan et al. [46] proposed applying a local search procedure to only solutions that are located apart from others. In addition, the neighborhood size in the local search depends on the density or crowdedness of solutions. Being selective in applying a local search, this strategy is computationally efficient and also aims to main diversity.

5. Multi-objective GA for Reliability Optimization

Many engineering problems have multiple objectives, including engineering system design and reliability optimization. There have been several interesting and successful implementations of multi-objective GA for this class of problems. A few successful examples are described in the following paragraphs.

Marseguerra, Zio and Podofillini [33] determine optimal surveillance test intervals using multi-objective GA with the goal of improving reliability and availability. Their research implemented a multi-objective GA which transparently and explicitly accounts for the uncertainties in the parameters. The objectives considered were the inverse of the expected system failure probability and the inverse of its variance. These are used to drive the genetic search toward solutions which are guaranteed to give optimal performance with high assurance, i.e., low estimation variance. They successfully applied their procedure to a complex system, a residual heat removal safety system for a boiling water reactor.

Martorell et al. [35] studied the selection of technical specifications and maintenance activities at nuclear power plants to increase reliability, availability and maintainability (RAM) of safety-related equipment. However, to improve RAM, additional limited resources (e.g. costs, task force, etc.) are required creating a multi-objective problem. They demonstrated the viability and significance of their proposed approach using multi-objective GA for an emergency diesel generator system.

Additionally, Martorell et al. [34] considered the optimal allocation of more reliable equipment, testing and maintenance activities to assure high RAM levels for safety-related systems. For these problems, the decision-maker encounters a multi-objective optimization problem where the parameters of design, testing and maintenance are decision variables. Solutions were obtained by using both single-objective GA and multi-objective GA, which were demonstrated to solve the problem of testing and maintenance optimization based on unavailability and cost criteria.

Sasaki and Gen [43] introduce a multi-objective problem which had fuzzy multiple objective functions and constraints with GUB (Generalized Upper Bounding) structure. They solved this problem by using a new hybridized GA. This approach leads to a flexible optimal system design by applying fuzzy goals and fuzzy constraints. A new chromosome representation was introduced in their work. To demonstrate the effectiveness of their method, a large-scale optimal system reliability design problem was analyzed.

Reliability allocation to minimize total plant costs, subject to an overall plant safety goal, is presented by Yang [52]. For their problem, design optimization is needed to improve the design, operation and safety of new and/or existing nuclear power plants. They presented an approach to determine the reliability characteristics of reactor systems, subsystems, major components and plant procedures that are consistent with a set of top-level performance goals. To optimize the reliability of the system, the cost for improving and/or degrading the reliability of the system are also included in the reliability allocation process creating a multi-objective problem. GA was applied to the reliability allocation problem of a typical pressurized water reactor.

Elegbede and Adjallah [10] present a methodology to optimize the availability and the cost of repairable parallel-series systems. It is a multi-objective combinatorial optimization, modeled with continuous and discrete variables. They transform the problem into a single objective problem and used traditional GA.

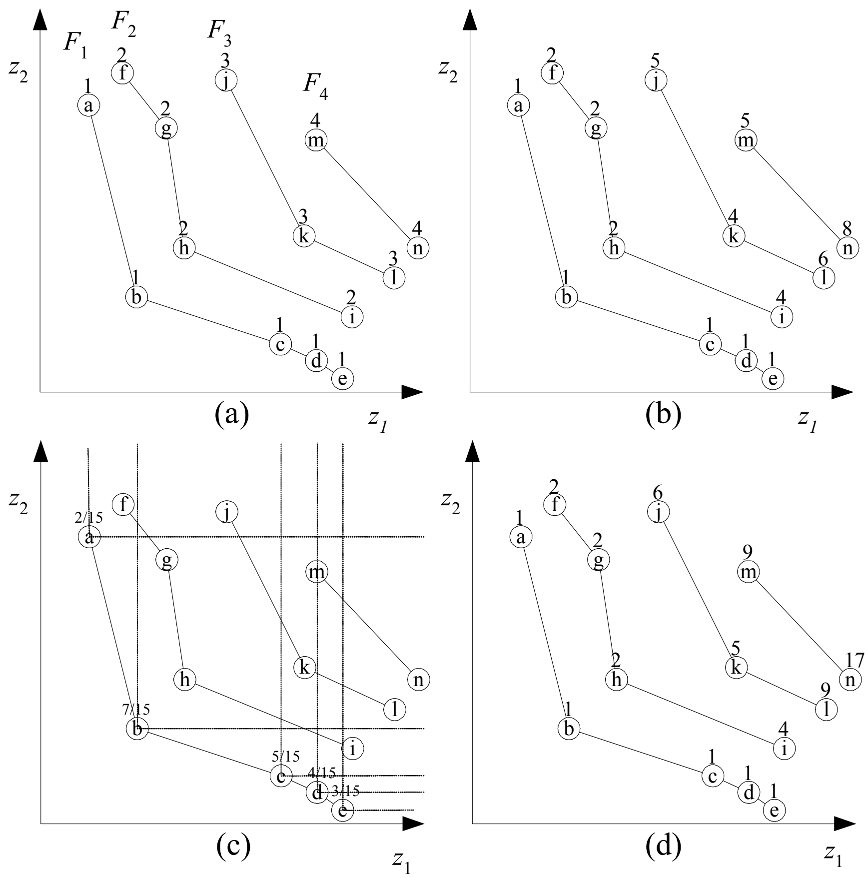

Figure 1. Ranking methods used in multi-objective GA.

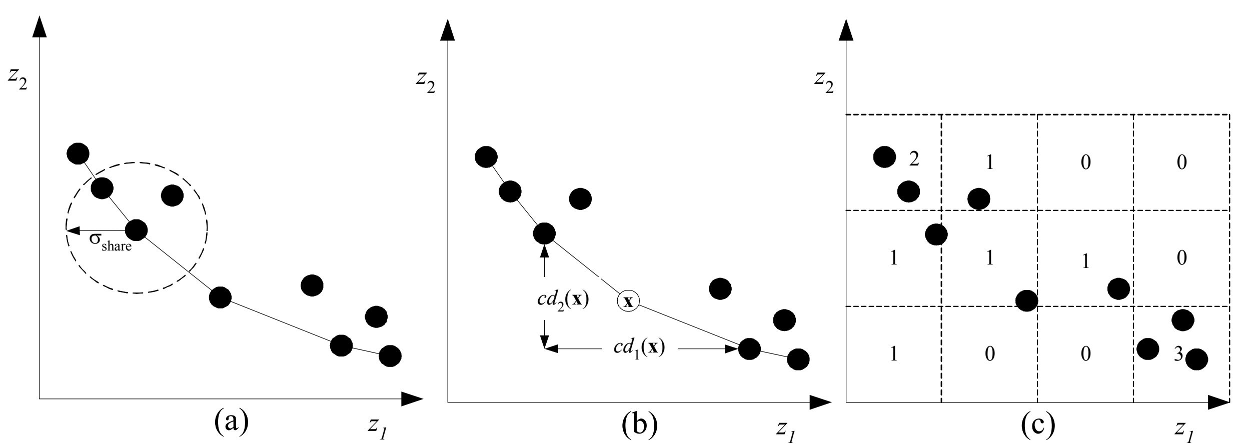

Figure 2. Diversity methods used in multi-objective GA.

REFERENCES

[1] Coello, C.A.C., A comprehensive survey of evolutionary-based multiobjective optimization techniques, *Knowledge and Information Systems* 1(3) (1999) 269-308.

[2] Coello, C.A.C. An updated survey of evolutionary multiobjective optimization techniques: state of the art and future trends. in *Proceedings of the 1999. Congress on Evolutionary Computation-CEC99, 6-9 July 1999*. 1999. Washington, DC, USA: IEEE.

[3] Coello, C.A.C., An updated survey of GA-based multiobjective optimization techniques, *ACM Computing Surveys* 32(2) (2000) 109-143.

[4] Coello Coello, C., List of References on Evolutionary Multiobjective Optimization, http://www.lania.mx/~ccoello/EMOO/EMOObib.html

[5] de Toro, F., Ortega, J., Fernandez, J., and Diaz, A. PSFGA: a parallel genetic algorithm for multiobjective optimization. in *Proceedings 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing, 9-11 Jan. 2002*. 2002. Canary Islands, Spain: IEEE Comput. Soc.

[6] Deb, K., Multi-Objective Optimization using Evolutionary Algorithms, John Wiley & Sons, Ltd, 2001.

[7] Deb, K. and Goel, T. A hybrid multi-objective evolutionary approach to engineering shape design. in *Evolutionary Multi-Criterion Optimization. First International Conference, EMO 2001, 7-9 March 2001*. 2001. Zurich, Switzerland: Springer-Verlag.

[8] Deb, K. and Goldberg, D.E. An investigation of of niche an species fromation in genetic function optimization. in *The Third International Conference on Genetic Algorithms*. 1989. George Mason University.

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

[10] Elegbede, C. and Adjallah, K., Availability allocation to repairable systems with genetic algorithms: A multi-objective formulation, *Reliability Engineering and System Safety* 82(3) (2003) 319-330.

[11] Fieldsend, J.E., Everson, R.M., and Singh, S., Using unconstrained elite archives for multiobjective optimization, *IEEE Transactions on Evolutionary Computation* 7(3) (2003) 305-323.

[12] Fonseca, C.M. and Fleming, P.J. Genetic algorithms for multiobjective optimization: formulation, discussion and generalization. in *Proceedings of ICGA-93: Fifth International Conference on Genetic Algorithms, 17-22 July 1993*. 1993. Urbana-Champaign, IL, USA: Morgan Kaufmann.

[13] Fonseca, C.M. and Fleming, P.J. Multiobjective genetic algorithms. in *IEE Colloquium on `Genetic Algorithms for Control Systems Engineering' (Digest No. 1993/130), 28 May 1993*. 1993. London, UK: IEE.

[14] Fonseca, C.M. and Fleming, P.J., Multiobjective optimization and multiple constraint handling with evolutionary algorithms. I. A unified formulation, *IEEE Transactions on Systems, Man & Cybernetics, Part A (Systems & Humans)* 28(1) (1998) 26-37.

[15] Goldberg, D.E., Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, MA, 1989.

[16] Goldberg, D.E. and Richardson, J. Genetic algorithms with sharing for multimodal function optimization. in *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms, 28-31 July 1987*. 1987. Cambridge, MA, USA: Lawrence Erlbaum Associates.

[17] Hajela, P. and Lin, C.-Y., Genetic search strategies in multicriterion optimal design, *Structural Optimization* 4(2) (1992) 99-107.

[18] Holland, J.H., Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, 1975.

[19] Horn, J., Nafpliotis, N., and Goldberg, D.E. A niched Pareto genetic algorithm for multiobjective optimization. in *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence, 27-29 June 1994*. 1994. Orlando, FL, USA: IEEE.

[20] Ishibuchi, H. and Murata, T. Multi-objective genetic local search algorithm. in *Proceedings of IEEE International Conference on Evolutionary Computation, 20-22 May 1996*. 1996. Nagoya, Japan: IEEE.

[21] Ishibuchi, H., Yoshida, T., and Murata, T., Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling, *IEEE Transactions on Evolutionary Computation* 7(2) (2003) 204-223.

[22] Jensen, M.T., Reducing the run-time complexity of multiobjective EAs: The NSGA-II and other algorithms, *IEEE Transactions on Evolutionary Computation* 7(5) (2003) 503-515.

[23] Jimenez, F., Gomez-Skarmeta, A.F., Sanchez, G., and Deb, K. An evolutionary algorithm for constrained multi-objective optimization. in *Proceedings of 2002 World Congress on Computational Intelligence - WCCI'02, 12-17 May 2002*. 2002. Honolulu, HI, USA: IEEE.

[24] Jimenez, F., Verdegay, J.L., and Gomez-Skarmeta, A.F. Evolutionary techniques for constrained multiobjective optimization problems. in *Workshop on Multi-Criterion Optimization Using Evolutionary Methods GECCO-1999*. 1999.

[25] Jones, D.F., Mirrazavi, S.K., and Tamiz, M., Multiobjective meta-heuristics: an overview of the current state-of-the-art, *European Journal of Operational Research* 137(1) (2002) 1-9.

[26] Knowles, J. and Corne, D. The Pareto archived evolution strategy: a new baseline algorithm for Pareto multiobjective optimisation. in *Proceedings of the 1999. Congress on Evolutionary Computation-CEC99, 6-9 July 1999*. 1999. Washington, DC, USA: IEEE.

[27] Knowles, J.D. and Corne, D.W., Approximating the nondominated front using the Pareto archived evolution strategy, *Evolutionary Computation* 8(2) 149-172.

[28] Knowles, J.D. and Corne, D.W. M-PAES: a memetic algorithm for multiobjective optimization. in *Proceedings of 2000 Congress on Evolutionary Computation, 16-19 July 2000*. 2000. La Jolla, CA, USA: IEEE.

[29] Konak, A. and Smith, A.E. Multiobjective optimization of survivable networks considering reliability. in *The 10th International Conference on Telecommunication Systems*. 2002. Naval Postgraduate School, Monterey, CA,.

[30] Konak, A. and Smith, A.E., Capacitated Network Design Considering Survivability: An Evolutionary Approach, *Journal of Engineering Optimization* 36(2) (2004) 189-205.

[31] Kursawe, F. A variant of evolution strategies for vector optimization. in *Parallel Problem Solving from Nature. 1st Workshop, PPSN 1 Proceedings, 1-3 Oct. 1990*. 1991. Dortmund, West Germany: Springer-Verlag.

[32] Lu, H. and Yen, G.G., Rank-density-based multiobjective genetic algorithm and benchmark test function study, *IEEE Transactions on Evolutionary Computation* 7(4) (2003) 325-343.

[33] Marseguerra, M., Zio, E., and Podofillini, L., Optimal reliability/availability of uncertain systems via multi-objective genetic algorithms, *IEEE Transactions on Reliability* 53(3) 424-434.

[34] Martorell, S., Sanchez, A., Carlos, S., and Serradell, V., Alternatives and challenges in optimizing industrial safety using genetic algorithms, *Reliability Engineering & System Safety* 86(1) (2004) 25-38.

[35] Martorell, S., Villanueva, J.F., Carlos, S., Nebot, Y., Sanchez, A., Pitarch, J.L., and Serradell, V., RAMS+C informed decision-making with application to multi-objective optimization of technical specifications and maintenance using genetic algorithms, *Reliability Engineering and System Safety* 87(1) (2005) 65-75.

[36] Miller, B.L. and Shaw, M.J. Genetic algorithms with dynamic niche sharing for multimodal function optimization. in *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation, ICEC'96, May 20-22 1996*. 1996. Nagoya, Jpn: IEEE, Piscataway, NJ, USA.

[37] Morse, J.N., Reducing the size of the nondominated set: pruning by clustering, *Computers & Operations Research* 7(1-2) (1980) 55-66.

[38] Mostaghim, S., Teich, J., and Tyagi, A. Comparison of data structures for storing Pareto-sets in MOEAs. in *Proceedings of 2002 World Congress on Computational Intelligence - WCCI'02, 12-17 May 2002*. 2002. Honolulu, HI, USA: IEEE.

[39] Murata, T. and Ishibuchi, H. MOGA: multi-objective genetic algorithms. in *Proceedings of 1995 IEEE International Conference on Evolutionary Computation, 29 Nov.-1 Dec. 1995*. 1995. Perth, WA, Australia: IEEE.

[40] Murata, T., Ishibuchi, H., and Tanaka, H., Multi-objective genetic algorithm and its applications to flowshop scheduling, *Computers & Industrial Engineering* 30(4) 957-968.

[41] Paquete, L. and Stutzle, T. A two-phase local search for the biobjective traveling salesman problem. in *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003. Proceedings, 8-11 April 2003*. 2003. Faro, Portugal: Springer-Verlag.

[42] Sarker, R., Liang, K.-H., and Newton, C., A new multiobjective evolutionary algorithm, *European Journal of Operational Research* 140(1) (2002) 12-23.

[43] Sasaki, M. and Gen, M., A method of fuzzy multi-objective nonlinear programming with GUB structure by Hybrid Genetic Algorithm, *International Journal of Smart Engineering System Design* 5(4) (2003) 281-288.

[44] Schaffer, J.D. Multiple Objective optimization with vector evaluated genetic algorithms. in *International Conference on Genetic Algorithm and their applications*. 1985.

[45] Srinivas, N. and Deb, K., Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms, *Journal of Evolutionary Computation* 2(3) (1994) 221-248.

[46] Tan, K.C., Lee, T.H., and Khor, E.F., Evolutionary algorithms with dynamic population size and local exploration for multiobjective optimization, *IEEE Transactions on Evolutionary Computation* 5(6) (2001) 565-588.

[47] Van Veldhuizen, D.A. and Lamont, G.B., Multiobjective evolutionary algorithms: analyzing the state-of-the-art, *Evolutionary Computation* 8(2) 125-147.

[48] Van Veldhuizen, D.A., Zydallis, J.B., and Lamont, G.B., Considerations in engineering parallel multiobjective evolutionary algorithms, *IEEE Transactions on Evolutionary Computation* 7(2) (2003) 144-173.

[49] Wilson, L.A., Moore, M.D., Picarazzi, J.P., and Miquel, S.D.S. Parallel genetic algorithm for search and constrained multi-objective optimization. in *Proceedings. 18th International Parallel and Distributed Processing Symposium, 26-30 April 2004*. 2004. Santa Fe, NM, USA: IEEE Comput. Soc.

[50] Xiong, S. and Li, F. Parallel strength Pareto multiobjective evolutionary algorithm. in *Proceedings of the Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies, 27-29 Aug. 2003*. 2003. Chengdu, China: IEEE.

[51] Xiujuan, L. and Zhongke, S., Overview of multi-objective optimization methods, *Journal of Systems Engineering and Electronics* 15(2) (2004) 142-146.

[52] Yang, J.-E., Hwang, M.-J., Sung, T.-Y., and Jin, Y., Application of genetic algorithm for reliability allocation in nuclear power plants, *Reliability Engineering & System Safety* 65(3) 229-238.

[53] Yen, G.G. and Lu, H., Dynamic multiobjective evolutionary algorithm: adaptive cell-based rank and density estimation, *IEEE Transactions on Evolutionary Computation* 7(3) (2003) 253-274.

[54] Zitzler, E., Deb, K., and Thiele, L., Comparison of multiobjective evolutionary algorithms: empirical results, *Evolutionary Computation* 8(2) 173-195.

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