Available online at www.sciencedirect.com
SCIENCE DIRECT®
Reliability Engineering and System Safety 91 (2006) 992–1007
RELIABILITY ENGINEERING & SYSTEM SAFETY
www.elsevier.com/locate/ress

Multi-objective optimization using genetic algorithms: A tutorial

Abdullah Konak[a,*], David W. Coit[b], Alice E. Smith[c]

[a]*Information Sciences and Technology, Penn State Berks, USA*
[b]*Department of Industrial and Systems Engineering, Rutgers University*
[c]*Department of Industrial and Systems Engineering, Auburn University*

Available online 9 January 2006

Abstract

Multi-objective formulations are realistic models for many complex engineering optimization 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 (GA) developed specifically for problems with multiple objectives. They differ primarily from traditional GA by using specialized fitness functions and introducing methods to promote solution diversity.

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 and introducing methods to promote solution diversity.

There are two general approaches to multiple-objective optimization. One is to combine the individual objective functions into a single composite function or move all but one objective to the constraint set. In the former case, determination of a single objective is possible with methods such as utility theory, weighted sum method, etc., but the problem lies in the proper selection of the weights or utility functions to characterize the decision-maker's preferences. In practice, it can be very difficult to precisely and accurately select these weights, even for someone familiar with the problem domain. Compounding this drawback is that scaling amongst objectives is needed and small perturbations in the weights can sometimes lead to quite different solutions. In the latter case, the problem is that to move objectives to the constraint set, a constraining value must be established for each of these former objectives. This can be rather arbitrary. In both cases, an optimization method would return a single solution rather than a set of solutions that can be examined for trade-offs. For this reason, decision-makers often prefer a set of good solutions considering 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(s) to achieve a certain amount of gain in the other(s). Pareto optimal solution sets are often preferred to single solutions because they can be practical when considering real-life problems

*Corresponding author.
E-mail address: konak@psu.edu (A. Konak).

0951-8320/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.ress.2005.11.018

since the final solution of the decision-maker is always a trade-off. Pareto optimal sets can be of varied sizes, but the size of the Pareto set usually increases with the increase in the number of objectives.

2. Multi-objective optimization formulation

Consider a decision-maker who wishes to optimize K objectives such that the objectives are non-commensurable and the decision-maker has no clear preference of the objectives relative to each other. Without loss of generality, all objectives are of the minimization type—a minimization type objective can be converted to a maximization type by multiplying negative one. A minimization multi-objective decision problem with K objectives 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 $z(x^*) = \{z_1(\mathbf{x}^*),\ldots,z_K(\mathbf{x}^*)\}$. 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, $z_i(\mathbf{x}) \leqslant z_i(\mathbf{y})$ 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 $\mathbf{X}$ is referred to as the *Pareto optimal set,* and for a given Pareto optimal set, the corresponding objective function values in the objective space are called the *Pareto front*. For many problems, the number of Pareto optimal solutions is enormous (perhaps 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 well as possible. With these concerns in mind, a multi-objective optimization approach should achieve the following three conflicting goals [1]:

1. The best-known Pareto front should be as close as possible 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. 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.

For a given computational time limit, the first goal is best served by focusing (intensifying) the search on a particular region of the Pareto front. On the contrary, the second goal demands the search effort to be uniformly distributed over the Pareto front. The third goal aims at extending the Pareto front at both ends, exploring new extreme solutions.

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

3. Genetic algorithms

The concept of GA was developed by Holland and his colleagues in the 1960s and 1970s [2]. GA are 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 digits. 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 work on the *encoding* of a problem, not on the problem itself.

GA operate 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 and depends on the length of the chromosome. 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 [3] 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 on its objective function value and infeasibility.
Step 5: *Selection:* Select N solutions from Q_t based on their fitness and copy them to P_{t+1}.
Step 6: If the stopping criterion is satisfied, terminate the search and return to the current population, else, set $t = t + 1$ go to Step 2.

4. Multi-objective GA

Being a population-based approach, GA are well suited to solve multi-objective optimization problems. A generic single-objective GA can be 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 have been the most popular heuristic approach to multi-objective design and optimization problems. Jones et al. [4] 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 GA (or VEGA), was proposed by Schaffer [5]. Afterwards, several multi-objective evolutionary algorithms were developed including Multi-objective Genetic Algorithm (MOGA) [6], Niched Pareto Genetic Algorithm (NPGA) [7], Weight-based Genetic Algorithm (WBGA) [8], Random Weighted Genetic Algorithm (RWGA)[9], Nondominated Sorting Genetic Algorithm (NSGA) [10], Strength Pareto Evolutionary Algorithm (SPEA) [11], improved SPEA (SPEA2) [12], Pareto-Archived Evolution Strategy (PAES) [13], Pareto Envelope-based Selection Algorithm (PESA) [14], Region-based Selection in Evolutionary Multiobjective Optimization (PESA-II) [15], Fast Non-dominated Sorting Genetic Algorithm (NSGA-II) [16], Multi-objective Evolutionary Algorithm (MEA) [17], Micro-GA [18], Rank-Density Based Genetic Algorithm (RDGA) [19], and Dynamic Multi-objective Evolutionary Algorithm (DMOEA) [20]. 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 [1,11,21–27] have been published on evolutionary multi-objective optimization. Coello lists more than 2000 references in his website [28]. Generally, multi-objective GA differ based on their fitness assignment procedure, elitisim, or diversification approaches. In Table 1, highlights of the well-known multi-objective with their advantages and disadvantages are given. 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

Table 1
A list of well-known multi-objective GA

Algorithm	Fitness assignment	Diversity mechanism	Elitism	External population	Advantages	Disadvantages
VEGA [5]	Each subpopulation is evaluated with respect to a different objective	No	No	No	First MOGA Straightforward implementation	Tend converge to the extreme of each objective
MOGA [6]	Pareto ranking	Fitness sharing by niching	No	No	Simple extension of single objective GA	Usually slow convergence Problems related to niche size parameter
WBGA [8]	Weighted average of normalized objectives	Niching Predefined weights	No	No	Simple extension of single objective GA	Difficulties in nonconvex objective function space
NPGA [7]	No fitness assignment, tournament selection	Niche count as tie-breaker in tournament selection	No	No	Very simple selection process with tournament selection	Problems related to niche size parameter Extra parameter for tournament selection
RWGA [9]	Weighted average of normalized objectives	Randomly assigned weights	Yes	Yes	Efficient and easy implement	Difficulties in nonconvex objective function space
PESA [14]	No fitness assignment	Cell-based density	Pure elitist	Yes	Easy to implement Computationally efficient	Performance depends on cell sizes Prior information needed about objective space
PAES [29]	Pareto dominance is used to replace a parent if offspring dominates	Cell-based density as tie breaker between offspring and parent	Yes	Yes	Random mutation hill-climbing strategy Easy to implement Computationally efficient	Not a population based approach Performance depends on cell sizes
NSGA [10]	Ranking based on non-domination sorting	Fitness sharing by niching	No	No	Fast convergence	Problems related to niche size parameter
NSGA-II [30]	Ranking based on non-domination sorting	Crowding distance	Yes	No	Single parameter (N) Well tested Efficient	Crowding distance works in objective space only
SPEA [11]	Raking based on the external archive of non-dominated solutions	Clustering to truncate external population	Yes	Yes	Well tested No parameter for clustering	Complex clustering algorithm
SPEA-2 [12]	Strength of dominators	Density based on the k-th nearest neighbor	Yes	Yes	Improved SPEA Make sure extreme points are preserved	Computationally expensive fitness and density calculation
RDGA [19]	The problem reduced to bi-objective problem with solution rank and density as objectives	Forbidden region cell-based density	Yes	Yes	Dynamic cell update Robust with respect to the number of objectives	More difficult to implement than others
DMOEA [20]	Cell-based ranking	Adaptive cell-based density	Yes (implicitly)	No	Includes efficient techniques to update cell densities Adaptive approaches to set GA parameters	More difficult to implement than others

goals in multi-objective optimization. This approach is also taken in the survey paper by Zitzler et al. [1]. However, the discussion in this paper is aimed at introducing the components of multi-objective GA to researchers and practitioners without a background on the multi-objective GA. It is also import to note that although several of the state-of-the-art algorithms exist as cited above, many researchers that applied multi-objective GA to their problems have preferred to design their own customized algorithms by adapting strategies from various multi-objective GA. This observation is another motivation for introducing the components of multi-objective GA rather than focusing on several algorithms. However, the pseudo-code for some of the well-known multi-objective GA are also provided in order to demonstrate how these procedures are incorporated within a multi-objective GA.

5. Design issues and components of multi-objective GA

5.1. Fitness functions

5.1.1. 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:

$$\min z = w_1 z'_1(\mathbf{x}) + w_2 z'_2(\mathbf{x}) + \cdots + w_k z'_k(\mathbf{x}), \quad (1)$$

where $z'_i(\mathbf{x})$ is the normalized objective function $z_i(\mathbf{x})$ and $\sum w_i = 1$. This approach is called the 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 must 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 [8] proposed the WBGA 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 the summed 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 [9,31] have proposed a MOGA 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 additional parameters. The general procedure of the RWGA using random weights is given as follows [31]:

Procedure RWGA:

E = external archive to store non-dominated solutions found during the search so far;

n_E = number of elitist solutions immigrating from E to P in each generation.

Step 1: Generate a random population.

Step 2: Assign a fitness value to each solution $\mathbf{x} \in P_t$ by performing the following steps:

Step 2.1: Generate a random number u_k in [0,1] for each objective k, $k = 1, \ldots, K$.

Step 2.2: Calculate the random weight of each objective k as $w_k = (1/u_k)\sum_{i=1}^{K} u_i$.

Step 2.3: Calculate the fitness of the solution as $f(\mathbf{x}) = \sum_{k=1}^{K} w_k z_k(\mathbf{x})$.

Step 3: Calculate the selection probability of each solution $\mathbf{x} \in P_t$ as follows: $p(\mathbf{x}) = (f(\mathbf{x}) - f^{\min})^{-1} \sum_{\mathbf{y} \in P_t} (f(\mathbf{y}) - f^{\min})$ where $f^{\min} = \min\{f(\mathbf{x}) | \mathbf{x} \in P_t\}$.

Step 4: Select parents using the selection probabilities calculated in Step 3. Apply crossover on the selected parent pairs to create N offspring. Mutate offspring with a predefined mutation rate. Copy all offspring to P_{t+1}. Update E if necessary.

Step 5: Randomly remove n_E solutions from P_{t+1} and add the same number of solutions from E to P_{t+1}.

Step 6: If the stopping condition is not satisfied, set $t = t + 1$ and go to Step 2. Otherwise, return to E.

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 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, multi-objective GA based on the weighed sum approach have difficulty in finding solutions uniformly distributed over a non-convex trade-off surface [1].

5.1.2. Altering objective functions

As mentioned earlier, VEGA [5] is the first GA used to approximate the Pareto-optimal set by a set of non-dominated solutions. In VEGA, population P_t is randomly divided into K equal sized sub-populations; $P_1, P_2, \ldots, P_K$. 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 as for a single objective GA.

Procedure VEGA:

N_S = subpopulation size ($N_S = N/K$)

Step 1: Start with a random initial population P_0. Set $t = 0$.

Step 2: If the stopping criterion is satisfied, return P_t.

Step 3: Randomly sort population P_t.

Step 4: For each objective k, $k = 1, \ldots K$, perform the following steps:

Step 4.1: For $i = 1 + (k-1)N_S, \ldots, kN_S$, assign fitness value $f(\mathbf{x}_i) = z_k(\mathbf{x}_i)$ to the ith solution in the sorted population.

Step 4.2: Based on the fitness values assigned in Step 4.1, select N_S solutions between the $(1 + (k-1)N_S)$th and (kN_S)th solutions of the sorted population to create subpopulation P_k.

Step 5: Combine all subpopulations $P_1, \ldots, P_k$ and apply crossover and mutation on the combined population to create P_{t+1} of size N. Set $t = t + 1$, go to Step 2.

A similar approach to VEGA is to use only a single objective function which is randomly determined each time in the selection phase [32]. The main advantage of the alternating objectives approach is easy to implement and

computationally as efficient as a single-objective GA. In fact, this approach is a straightforward extension of a single objective GA to solve multi-objective problems. The major drawback of objective switching is that the population tends to converge to solutions which are superior in one objective, but poor at others.

5.1.3. 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 [3] 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 $TP = TP - F_i$. If $TP = \varnothing$ 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. NSGA [10] also classifies the population into non-dominated fronts using an algorithm similar to that given above. Then a dummy fitness value is assigned to each front using a fitness sharing function such that the worst fitness value assigned to F_i is better than the best fitness value assigned to F_{i+1}. NSGA-II [16], a more efficient algorithm, named the fast non-dominated-sort algorithm, was developed to form non-dominated fronts. Fonseca and Fleming [6] used a slightly different rank assignment approach than the ranking based on non-dominated-fronts as follows:

$$r_2(\mathbf{x}, t) = 1 + nq(\mathbf{x}, t), \tag{2}$$

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 Fig. 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.

SPEA [11] uses a ranking procedure to assign better fitness values to non-dominated solutions at underrepresented regions of the objective space. In 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

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

where $\mathrm{np}(\mathbf{y}, t)$ 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

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

Fig. 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 [19] 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 $\mathbf{x}$, the rank of the solutions dominating this solution must be calculated first. Fig. 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**.

Although some of the ranking approaches described in this section can be used directly to assign fitness values to individual solutions, they are usually combined with various fitness sharing techniques to achieve the second goal in multi-objective optimization, finding a diverse and uniform Pareto front.

5.2. 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 Pareto front. Without taking preventive measures, the population tends to form relatively few clusters in multi-objective GA. This phenomenon is called genetic drift, and several approaches have been devised to prevent genetic drift as follows.

5.2.1. Fitness sharing

Fitness sharing encourages 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 penalty

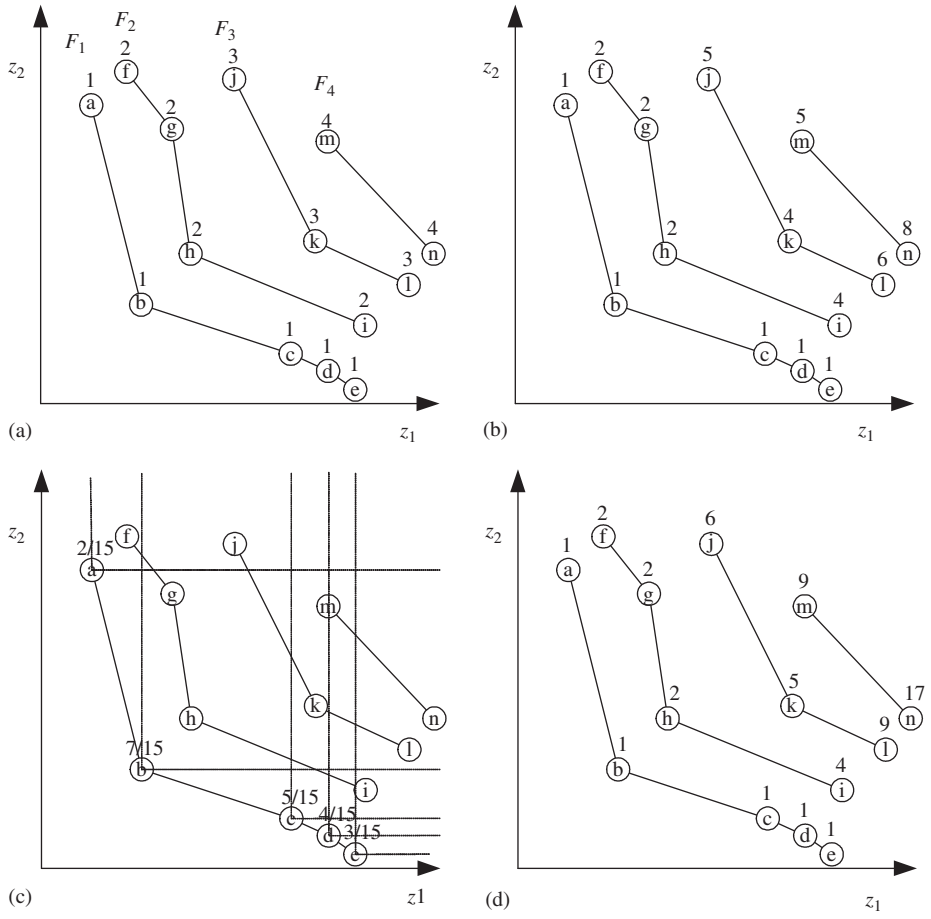

Fig. 1. Ranking methods used in multi-objective GA.

method is used to penalize the solutions located in such areas.

The idea of fitness sharing was first proposed by Goldberg and Richardson [33] in the investigation of multiple local optima for multi-modal functions. Fonseca and Fleming [6] used this idea to penalize clustered solutions with the same rank as follows:

Step 1: Calculate the Euclidean distance between every solution pair $\mathbf{x}$ and $\mathbf{y}$ in the normalized objective space between 0 and 1 as

$$dz(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{k=1}^{K}\left(\frac{z_k(\mathbf{x}) - z_k(\mathbf{y})}{z_k^{\max} - z_k^{\min}}\right)^2}, \tag{3}$$

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

$$nc(\mathbf{x},t) = \sum_{\mathbf{y}\in P, r(\mathbf{y},t)=r(\mathbf{x},t)} \max\left\{\frac{\sigma_{\text{share}} - dz(\mathbf{x},\mathbf{y})}{\sigma_{\text{share}}}, 0\right\}, \tag{4}$$

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 (Fig. 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 distance in the decision variable space between two solutions $\mathbf{x}$ and $\mathbf{y}$ which is defined as

$$dx(\mathbf{x},\mathbf{y}) = \sqrt{\frac{1}{M}\sum_{i=1}^{M}(x_i - y_i)^2} \tag{5}$$

in the calculation of niche count. Eq. (5) is a measure of the 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 [34] reported that fitness sharing in objective function space usually performs better than one based on decision variable space.

One of the disadvantages of 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 [34] and Fonseca and Fleming [6] 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 usually surpass the cost of extra computational effort. Miller and Shaw [35] proposed a dynamic niche sharing approach to increase effectiveness of computing niche counts.

MOGA [6] was the first multi-objective GA that explicitly used Pareto-based ranking and niching techniques together to encourage the search toward the true Pareto front while maintaining diversity in the population. Therefore, it is a good example to demonstrate how Pareto-based ranking and fitness sharing can be integrated in a multi-objective GA. The procedure of the MOGA is given as follows:

Procedure MOGA:

Step 1: Start with a random initial population P_0. Set $t = 0$.
Step 2: If the stopping criterion is satisfied, return P_t.
Step 3: Evaluate fitness of the population as follows:
Step 3.1: Assign a rank $r(\mathbf{x},t)$ to each solution $\mathbf{x} \in P_t$ using the ranking scheme given in Eq. (2).
Step 3.2: Assign a fitness values to each solution based on the solution's rank as follows [36]:

$$f(\mathbf{x},t) = N - \sum_{k=1}^{r(\mathbf{x},t)-1} n_k - .5 \times (n_{r(\mathbf{x},t)} - 1)$$

where n_k is the number of the solutions with rank k.
Step 3.3: Calculate the niche count $\text{nc}(\mathbf{x},t)$ of each solution $\mathbf{x} \in P_t$ using Eq. (4).
Step 3.4: Calculate the shared fitness value of each solution $\mathbf{x} \in P_t$ as follows:

$$f'(\mathbf{x},t) = f(\mathbf{x},t)/\text{nc}(\mathbf{x},t).$$

Step 3.5: Normalize the fitness values by using the shared fitness values

$$f''(\mathbf{x},t) = \frac{f'(\mathbf{x},t) n_{r(\mathbf{x},t)}}{\sum_{\substack{\mathbf{y}\in P_t \\ r(\mathbf{y},t)=r(\mathbf{x},t)}} f'(\mathbf{x},t)} f(\mathbf{x},t).$$

Step 4: Use a stochastic selection method based on f'' to select parents for the mating pool. Apply crossover and mutation on the mating pool until offspring population Q_t of size N is filled. Set $P_{t+1} = Q_t$.
Step 5: Set $t = t + 1$, go to Step 2.

In SPEA2 [12], a density measure is used to discriminate between solutions with the same rank, where the density of a solution is defined as the inverse of the distance to its kth closest neighbor in objective function space. The density of a solution is similar to its niche count. However, selecting a value for parameter k is more straightforward then selecting a value for σ_{share}.

5.2.2. 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, NSGA-II [16] uses a crowding distance method as follows (Fig. 2b):

Step 1: Rank the population and identify non-dominated fronts $F_1, F_2, \ldots, F_R$. For each front $j = 1, \ldots, R$ 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 ith solution in the sorted list with respect to the objective function k. Assign $\text{cd}_k(\mathbf{x}_{[1,k]}) = \infty$ and $\text{cd}_k(\mathbf{x}_{[l,k]}) = \infty$, and for $i = 2, \ldots, l-1$ assign

$$\text{cd}_k(\mathbf{x}_{[i,k]}) = \frac{z_k(\mathbf{x}_{[i+1,k]}) - z_k(\mathbf{x}^k_{[i-1,k]})}{z_k^{\max} - z_k^{\min}}.$$

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

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 such as σ_{share} or the kth closest neighbor. In NSGA-II, this crowding distance measure is used as a tie-breaker in a selection technique called the crowded tournament selection operator: 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 is the winner. Otherwise, the solution with the lowest rank is selected.

5.2.3. Cell-based density

In this approach [13,19,20,29], the objective space is divided into K-dimensional cells (see Fig. 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 PESA [14], between two non-dominated solutions, the one with a lower density is preferable. The procedure of PESA is given as follows:

Procedure PESA:
N_E = the maximum size of non-dominated archive E,
N_P = the population size, n = number of the grids along each objective function axis.

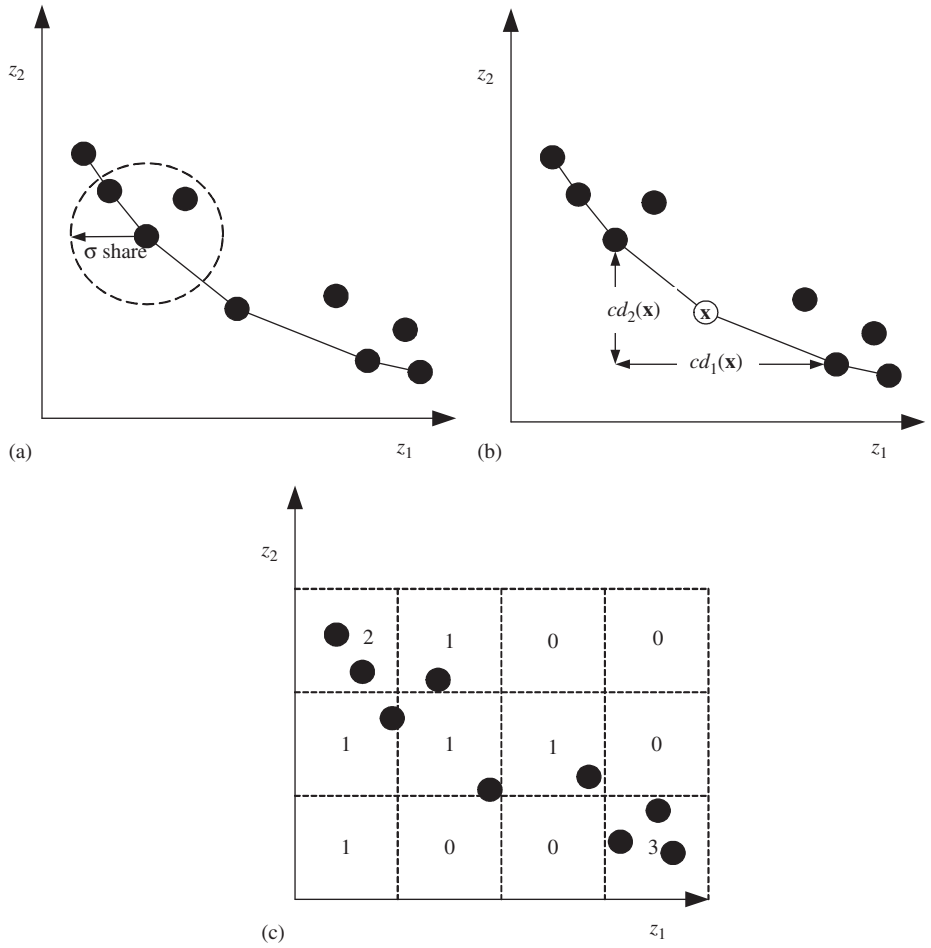

Fig. 2. Diversity methods used in multi-objective GA.

Step 1: Start with a random initial population P_0 and set external achieve $E_0 = \varnothing$, $t = 0$.
Step 2: Divide the normalized objective space into n^K hyper-cubes where n is the number of grids along a single objective axis and K is the number of objectives.
Step 3: Update non-dominated archive E_t by incorporating new solutions from P_t one by one as follows:

Case 1: If a new solution is dominated by at least a solution in E_t, discard the new solution.
Case 2: If a new solution dominates some solutions in E_t, remove those dominated solutions from E_t and add to the new solution to E_t. Update the membership of the hyper-cubes.
Case 3: If a new solution is not dominated by and does not dominate any solution in E_t, add this solution to E_t. If $| E_t | = N_E + 1$, randomly choose a solution from the most crowded hyper-cubes to be removed. Update the membership of the hyper-cubes.

Step 4: If the stopping criterion is satisfied, stop and return E_t.
Step 5: Set $P_t = \varnothing$, and select solutions from E_t for crossover and mutation based on the density information of the hyper-cubes. For example, if binary tournament selection is used, the winner is the solution located in the less crowded hyper-cubes. Apply crossover and mutation to generate N_P offspring and copy them to P_{t+1}.
Step 6: Set $t = t + 1$ and go to Step 3.

PESA-II [15] follows a more direct approach, namely region-based selection, where cells but not individual solutions are selected during the selection process. In this approach, a cell that is sparsely occupied has a higher chance to be selected than a crowded cell. Once a cell is selected, solutions within the cell are randomly chosen to participate to crossover and mutation.

Lu and Yen [19] and Yen and Lu [20] 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 to 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. RDGA [19] uses a cell-based density approach in an interesting way to convert a general K-objective problem into a bi-objective optimization

problem with the objectives to minimize the individual rank value and density of the population. The procedure of this approach is given as follows:

Procedure RDGA:

n_k = number of cells along the axis of objective function k.

Step 1: Create a random parent population P_0 of size N, $t = 0$.

Step 2: Divide the normalized objective space into $n_1 \times n_2 \times \cdots \times n_K$ hyper-cells.

Step 3: Update the cells dimensions as $d_k = (z_k^{\max} - z_k^{\min})/n_k$.

Step 4: Identify the cell membership of each solution $\mathbf{x} \in P_t$.

Step 5: Assign a density value each solution $\mathbf{x} \in P_t$ as $m(\mathbf{x}, t)$ =the number of solutions located in the same cell with **x.**

Step 6: Use the automatic accumulated tanking method to rank each solution as follows:

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

Step 7: Use the rank and density of each solution as the objectives of a bi-objective optimization problem. Use VEGA's fitness assignment approach to minimize the individual rank value and the population density while creating the mating pool. In addition, randomly copy solutions from the non-dominated archive to the mating pool.

Step 8: Apply crossover and mutation on the mating pool. A selected parent performs crossover only with the best solution in the parent's cell and neighborhood cells. Do not allow an offspring to be located in a cell dominated by its parents. Replace the selected parent if it is dominated by the offspring. Update non-dominated solutions archive. Set $t = t + 1$, go to Step 3 if the stopping condition is not satisfied.

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. RDGA [19] uses a method based on this global density map to push solutions out of high density areas towards low density areas. Another other advantage is its computational efficiency compared to the niching or neighborhood-based density techniques. Yen and Lu [20] proposed several data structures and algorithms to efficiently store cell information and modify cell densities.

5.3. Elitisim

Elitism in the context of single-objective GA means that the best solution found so far during the search always survives to the next generation. In this respect, all non-dominated solutions discovered by a multi-objective GA are considered 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. Early multi-objective GA did not use elitism. However, most recent multi-objective GA and their variations use elitism. As discussed in [11,36,37], multi-objective GA using elitist strategies tend to outperform their non-elitist counterparts. Multi-objective GA use two strategies to implement elitism [26]: (i) maintaining elitist solutions in the population, and (ii) storing elitist solutions in an external secondary list and reintroducing them to the population.

5.3.1. Strategies to maintain elitist solutions in the population

Random selection does not ensure that a non-dominated solution will survive to the next generation. A straightforward implementation of elitism in a multi-objective GA is to copy all non-dominated solutions 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 [38,39] proposed a multi-objective GA with a 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, Pareto domination tournament selection is used as follows [7]. 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 [17].

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 $|F_1 \cup F_2 \cup \cdots \cup F_k| \leqslant N$ and $|F_1 \cup F_2 \cup \cdots \cup F_k \cup F_{k+1}| > N$. 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 $|F_1| \leqslant N$, and the secondary selection based on crowding distance promotes diversity. The complete procedure of NSGA-II is given below to demonstrate an implementation of elitism without using a secondary external population.

Procedure NSGA-II:

Step 1: Create a random parent population P_0 of size N. Set $t = 0$.
Step 2: Apply crossover and mutation to P_0 to create offspring population Q_0 of size N.
Step 3: If the stopping criterion is satisfied, stop and return to P_t.
Step 4: Set $R_t = P_t \cup Q_t$.
Step 5: Using the fast non-dominated sorting algorithm, identify the non-dominated fronts F_1, F_2, …,F_k in R_t.
Step 6: For $i = 1,\ldots,k$ do following steps:
  Step 6.1: Calculate crowding distance of the solutions in F_i (as described in Section 5.2.2).
  Step 6.2: Create P_{t+1} as follows:
    Case 1: If $|P_{t+1}| + |F_i| \leqslant N$, then set $P_{t+1} = P_{t+1} \cup F_i$;
    Case 2: If $|P_{t+1}| + |F_i| > N$, then add the least crowded $N - |P_{t+1}|$ solutions from F_i to P_{t+1}.
Step 7: Use binary tournament selection based on the crowding distance to select parents from P_{t+1}. Apply crossover and mutation to P_{t+1} to create offspring population Q_{t+1} of size N.
Step 8: Set $t = t + 1$, and go to Step 3.

Note that when the combined parent and offspring population includes more N non-dominated solutions, NSGA-II becomes as a pure elitist GA where only non-dominated solutions participate in crossover and selection. The main advantage of maintaining non-dominated solutions in the population is straightforward implementation. In this strategy, the population size is an important GA parameter since no external archive is used to store discovered non-dominated solutions.

5.3.2. 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 identified so far during the search [11], and E is updated each time a new solution is created by removing elitist solutions dominated by a 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 have been proposed to efficiently store, update, and search in list E [40,41]. 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 have been proposed to control the size of E. For example, SPEA uses the average linkage clustering method [42] 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:

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

Here, the distance $\mathrm{d}(\mathbf{x}, \mathbf{y})$ can be calculated in objective function space using Eq. (3) or in decision variable space using Eq. (5).
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 $|C| \leqslant N$, 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 the 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 [11,19,20], solutions for P_{t+1} are selected from the combined population of P_t and E_t. To implement this strategy, populations P_t and E_t are combined, 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 room for n elitist solutions in the next population [43]. In this strategy, $N-n$ solutions are selected from parents and newly created offspring and n solutions are selected from E_t.

SPEA and SPEA2 are both very effective algorithms that use an external list to store non-dominated solution discovered so far in the search. They are also excellent examples for the use of external populations. The procedure of the SPEA2 is given as follows:

Procedure SPEA2:
N_E = the maximum size of the non-dominated archive E,
N_P = the population size,
k = parameter for density calculation $\left(k = \sqrt{N_E + N_P}\right)$.

Step 1: Randomly generate an initial solution P_0 and set $E_0 = \varnothing$.
Step 2: Calculate the fitness of each solution $\mathbf{x}$ in $P_t \cup E_t$ as follows:
  Step 2.1: $r(\mathbf{x}, t) = \sum_{\mathbf{y} \in P_t \cup E_t, \mathbf{y} \succ \mathbf{x}} s(\mathbf{y}, t)$ where $s(\mathbf{y}, t)$ is the number of solutions in $P_t \cup E_t$ dominated by solution $\mathbf{y}$.
  Step 2.2: Calculate the density as $m(\mathbf{x}, t) = (\sigma_{\mathbf{x}}^k + 1)^{-1}$ where $\sigma_{\mathbf{x}}^k$ is the distance between solution $\mathbf{x}$ and its kth nearest neighbor.
  Step 2.3: Assign a fitness value as $f(\mathbf{x}, t) = r(\mathbf{x}, t) + m(\mathbf{x}, t)$.
Step 3: Copy all non-dominated solutions in $P_t \cup E_t$ to E_{t+1}. Two cases are possible:
  Case 1: If $|E_{t+1}| > N_E$, then truncate $|E_{t+1}| - N_E$ solutions by iteratively removing solutions with the maximum σ^k distances. Break any tie by examining σ^l

for $l = k-1,\ldots,1$ sequentially.
Case 2: If $|E_{t+1}| \leqslant N_E$, copy the best $N_E - |E_{t+1}|$ dominated solutions according to their fitness values from $P_t \cup E_t$ to E_{t+1}.
Step 4: If the stopping criterion is satisfied, stop and return non-dominated solutions in E_{t+1}.
Step 5: Select parents from E_{t+1} using binary tournament selection with replacement.
Step 6: Apply crossover and mutation operators to the parents to create N_P offspring solutions. Copy offspring to P_{t+1}, $t = t+1$, and go to Step 2.

Other examples of elitist approaches using external populations are PESA [14], RDGA [44], RWGA [43], and DMOEA [20].

5.4. Constraint handling

Most real-world optimization problems include constraints that must be satisfied. An excellent survey on the constraint handling techniques used in evolutionary algorithms is given by Coello [45]. A single-objective GA may use one of four different constraint handling strategies: (i) discarding infeasible solutions (the "death penalty"); (ii) reducing the fitness of infeasible solutions by using a penalty function; (iii) crafting genetic operators to always produce feasible solutions; and (iv) transforming infeasible solutions to be feasible ("repair"). Handling of constraints has not been adequately researched for multi-objective GA [46]. For instance, all major multi-objective GA assume problems without constraints. While constraint handling strategies (i), (iii), and (iv) are directly applicable to the multi-objective case, implementation of penalty function strategies, which is the most frequently used constraint handling strategy in single-objective GA, is not straightforward in multi-objective GA due to fact that fitness assignment is based on the non-dominance rank of a solution, not on its objective function values.

Jimenez et al. [46,47] 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 those 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 count in decision variable space (Eq. (5)) is used for selection. In this case, the solution with the lower niche count is the winner.
Step 4: In the case that solutions **x** and **y** are both feasible, 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 is calculated in 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. The main disadvantages of this procedure are its computational complexity and additional parameters such as the size of reference set C and niche size. However, modifications are also possible. In Step 4, for example, the niche count of the solutions could be calculated in objective function space instead of decision variable space. In Step 3, the solution with the least infeasibility could be declared as the winner without comparing solutions **x** and **y** to a reference set C with respect to infeasibility. Such modifications could reduce the computational complexity of the procedure.

Deb et al. [16] 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 $F_1, F_2, F_3, \ldots, F_R$ are identified in a similar way to that defined in [3], 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 can be easily integrated into a multi-objective GA. A similar approach, called dominance-based tournament selection, was used by Coello and Montes [48] to solve single objective problems with several difficult constraints using a modified version of NPGA [7]. In this selection approach, dominance is defined with respect to the constraint violations of two solutions. Infeasible solution **x** is said to constrain-dominate **y** if it has fewer or equal constraint violations than solution **y** for every constraint of the problem, but less violation for at least one constraint. A tie between two constrain-non-dominated solutions is resolved by the total constraint violation of the solutions.

5.5. Parallel and hybrid multi-objective GA

All comparative studies on multi-objective GA agree that elitism and diversity preservation mechanisms improve performance. 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 requirements of multi-objective GA using advanced data structures. One of the latest trends is parallel and distributed processing. Several recent papers [49–52] presented parallel implementation of multi-objective GA over multiple processors.

Hybridization of GA with local search algorithms is frequently applied in single-objective GA. This approach is usually referred to as a memetic algorithm [53]. Generally, a local search algorithm proceeds as follows:

Step 1: Start with an initial solution **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 [54] described a bi-objective GA where local search is used to generate initial solutions by optimizing only one objective. Deb and Goel [55] applied local search to only final solutions. In Ishibuchi and Murata's approach [43], 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 et al. [56] also used the weighted sum of the objective functions to evaluate solutions 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 [53] presented a memetic version of PAES, called M-PAES. 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 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. 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. [57] proposed applying a local search procedure to only solutions that are located apart from others. In addition, the neighborhood size of the local search depends on the density or crowdedness of solutions. Being selective in applying a local search, this strategy is computationally efficient while maintaining diversity.

6. 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. These are described in the following paragraphs.

Marseguerra et al. [58] determined optimal surveillance test intervals using multi-objective GA with the goal of improving reliability and availability. Their research implemented a multi-objective GA which 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. [59] 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. money, work force) are required, posing 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. [60] 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 [61] introduced a multi-objective problem which had fuzzy multiple objective functions and constraints with a generalized upper bounding (GUB) 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, was presented by Yang et al. [62]. 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 was 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 [63] presented 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 transformed the problem into a single objective problem and used traditional GA.

Deb et al. [64] formulated a bi-objective optimization problem of minimizing total wire length and minimizing the failure rate in the printed circuit board design. They implemented NSGA-II to solve the problem. The results in the best Pareto fronts found were analyzed to understand the trade-offs between reliability and printed circuit board design.

Kumar et al. [65] presented a multi-objective GA approach to design telecommunication networks while simultaneously minimizing network performance and design costs under a reliability constraint.

7. Conclusions

Most real-world engineering problems involve simultaneously optimizing multi-objectives where considerations of trade-offs is important. In the last decade, evolutionary approaches have been the primary tools to solve real-world multi-objective problems. This paper presented multi-objective GA by focusing on their components and the salient issues encountered when implementing multi-objective GA.

Consideration of the computational realities along with the performance of the different methods is needed. Also, nearly all problems will require some customization of the GA approaches to properly handle the objectives, constraints, encodings and scale. It is envisioned that the Pareto solutions identified by GA would be pared down to a representative small set for the designer or engineer to further investigate. Therefore, for most implementations it is not vital to find every Pareto optimal solution, but rather, efficiently and reliably identify Pareto optimal solutions across the range of interest for each objective function.

In the reliability field, there are complicating factors—namely the usual computational effort require to evaluate or estimate reliability or availability metrics. This makes consideration of computational effort especially relevant and the user might need to define several options with differing effort (such as bounds, estimation, exact calculation) that are used for different phases of the search or for different solutions or regions. Also, control of the Pareto set size is critical to keep the computational effort to a reasonable level. While these aspects make the use of multiobjective GA more challenging in the reliability field, the method still offers the most promise of any with its powerful population-based search and its flexibility.

References

[1] Zitzler E, Deb K, Thiele L. Comparison of multiobjective evolutionary algorithms: empirical results. Evol Comput 2000;8(2):173–95.
[2] Holland JH. Adaptation in natural and artificial systems. Ann Arbor: University of Michigan Press; 1975.
[3] Goldberg DE. Genetic algorithms in search, optimization, and machine learning. Reading, MA: Addison-Wesley; 1989.
[4] Jones DF, Mirrazavi SK, Tamiz M. Multiobjective meta-heuristics: an overview of the current state-of-the-art. Eur J Oper Res 2002;137(1):1–9.
[5] Schaffer JD. Multiple objective optimization with vector evaluated genetic algorithms. In: Proceedings of the international conference on genetic algorithm and their applications, 1985.
[6] Fonseca CM, Fleming PJ. Multiobjective genetic algorithms. In: IEE colloquium on 'Genetic Algorithms for Control Systems Engineering' (Digest No. 1993/130), 28 May 1993. London, UK: IEE; 1993.
[7] 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, 27–29 June, 1994. Orlando, FL, USA: IEEE; 1994.
[8] Hajela P, lin C-y. Genetic search strategies in multicriterion optimal design. Struct Optimization 1992;4(2):99–107.
[9] Murata T, Ishibuchi H. MOGA: multi-objective genetic algorithms. In: Proceedings of the 1995 IEEE international conference on evolutionary computation, 29 November–1 December, 1995. Perth, WA, Australia: IEEE; 1995.
[10] Srinivas N, Deb K. Multiobjective optimization using nondominated sorting in genetic algorithms. J Evol Comput 1994;2(3):221–48.
[11] Zitzler E, Thiele L. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Trans Evol Comput 1999;3(4):257–71.

[12] Zitzler E, Laumanns M, Thiele L. SPEA2: improving the strength Pareto evolutionary algorithm. Swiss Federal Institute Techonology: Zurich, Switzerland; 2001.
[13] Knowles JD, Corne DW. Approximating the nondominated front using the Pareto archived evolution strategy. Evol Comput 2000;8(2):149–72.
[14] Corne DW, Knowles JD, Oates MJ. The Pareto envelope-based selection algorithm for multiobjective optimization. In: Proceedings of sixth international conference on parallel problem solving from Nature, 18–20 September, 2000. Paris, France: Springer; 2000.
[15] Corne D, Jerram NR, Knowles J, Oates J. PESA-II: region-based selection in evolutionary multiobjective optimization. In: Proceedings of the genetic and evolutionary computation conference (GECCO-2001), San Francisco, CA, 2001.
[16] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 2002;6(2):182–97.
[17] Sarker R, Liang K-H, Newton C. A new multiobjective evolutionary algorithm. Eur J Oper Res 2002;140(1):12–23.
[18] Coello CAC, Pulido GT. A micro-genetic algorithm for multiobjective optimization. In: Evolutionary multi-criterion optimization. First international conference, EMO 2001, 7–9 March, 2001. Zurich, Switzerland: Springer; 2001.
[19] Lu H, Yen GG. Rank-density-based multiobjective genetic algorithm and benchmark test function study. IEEE Trans Evol Comput 2003;7(4):325–43.
[20] Yen GG, Lu H. Dynamic multiobjective evolutionary algorithm: adaptive cell-based rank and density estimation. IEEE Trans Evol Comput 2003;7(3):253–74.
[21] Coello CAC. A comprehensive survey of evolutionary-based multiobjective optimization techniques. Knowl Inform Syst 1999;1(3):269–308.
[22] Coello CAC. 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. Washington, DC, USA: IEEE.
[23] Coello CAC. An updated survey of GA-based multiobjective optimization techniques. ACM Comput Surv 2000;32(2):109–43.
[24] Fonseca CM, Fleming PJ. Genetic algorithms for multiobjective optimization: formulation, discussion and generalization. In: Proceedings of the ICGA-93: fifth international conference on genetic algorithms, 17–22 July 1993. Urbana-Champaign, IL, USA: Morgan Kaufmann; 1993.
[25] Fonseca CM, Fleming PJ. Multiobjective optimization and multiple constraint handling with evolutionary algorithms. I. A unified formulation. IEEE Trans Syst Man Cybern A 1998;28(1):26–37.
[26] Jensen MT. Reducing the run-time complexity of multiobjective EAs: The NSGA-II and other algorithms. IEEE Trans Evol Comput 2003;7(5):503–15.
[27] Xiujuan L, Zhongke S. Overview of multi-objective optimization methods. J Syst Eng Electron 2004;15(2):142–6.
[28] Coello CAC. 2005, http://www.lania.mx/~ccoello/EMOO/EMOObib.html
[29] 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-CEC99, 6–9 July 1999. Washington, DC, USA: IEEE; 1999.
[30] Deb K, Agrawal S, Pratap A, Meyarivan T. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In: Proceedings of sixth international conference on parallel problem solving from nature, 18–20 September, 2000. Paris, France: Springer; 2000.
[31] Murata T, Ishibuchi H, Tanaka H. Multi-objective genetic algorithm and its applications to flowshop scheduling. Comput Ind Eng 1996;30(4):957–68.
[32] Kursawe F. A variant of evolution strategies for vector optimization. In: Parallel problem solving from nature. First workshop, PPSN 1 proceedings, 1–3 October, 1990. Dortmund, West Germany: Springer; 1991.
[33] Goldberg DE, 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. Cambridge, MA, USA: Lawrence Erlbaum Associates; 1987.
[34] Deb K, Goldberg DE. An investigation of of niche an species fromation in genetic function optimization. In: Proceedings of the third international conference on genetic algorithms, George Mason University, 1989.
[35] Miller BL, Shaw MJ. 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, Nagoya, Japan. Piscataway, NJ, USA: IEEE; 1996.
[36] Deb K. Multi-objective optimization using evolutionary algorithms. New York: Wiley; 2001.
[37] Van Veldhuizen DA, Lamont GB. Multiobjective evolutionary algorithms: analyzing the state-of-the-art. Evol Comput 2000;8(2):125–47.
[38] Konak A, Smith AE. Multiobjective optimization of survivable networks considering reliability. In: Proceedings of the 10th international conference on telecommunication systems. Monterey, CA: Naval Postgraduate School; 2002.
[39] Konak A, Smith AE. Capacitated network design considering survivability: an evolutionary approach. J Eng Optim 2004;36(2):189–205.
[40] Fieldsend JE, Everson RM, Singh S. Using unconstrained elite archives for multiobjective optimization. IEEE Trans Evol Comput 2003;7(3):305–23.
[41] Mostaghim S, Teich J, Tyagi A. Comparison of data structures for storing Pareto-sets in MOEAs. In: Proceedings of the 2002 world congress on computational intelligence—WCCI'02, 12–17 May, 2002. Honolulu, HI, USA: IEEE; 2002.
[42] Morse JN. Reducing the size of the nondominated set: pruning by clustering. Comput Oper Res 1980;7(1–2):55–66.
[43] Ishibuchi H, Murata T. Multi-objective genetic local search algorithm. In: Proceedings of the IEEE international conference on evolutionary computation, 20–22 May, 1996. Nagoya, Japan: IEEE; 1996.
[44] Lu H, Yen GG. Rank-density based multiobjective genetic algorithm. In: Proceedings of the 2002 world congress on computational intelligence—WCCI'02, 12–17 May, 2002. Honolulu, HI, USA: IEEE; 2002.
[45] Coello CAC. A survey of constraint handling techniques used with evolutionary algorithms. Veracruz, Mexico: Laboratorio Nacional de Informtica Avanzada; 1999.
[46] Jimenez F, Gomez-Skarmeta AF, Sanchez G, Deb K. An evolutionary algorithm for constrained multi-objective optimization. In: Proceedings of the 2002 world congress on computational intelligence—WCCI'02, 12–17 May, 2002. Honolulu, HI, USA: IEEE; 2002.
[47] Jimenez F, Verdegay JL, Gomez-Skarmeta AF. Evolutionary techniques for constrained multiobjective optimization problems. In: Workshop on multi-criterion optimization using evolutionary methods GECCO-1999, 1999.
[48] Coello CAC, Montes EM. Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. Adv Eng Inform 2002;16(3):193–203.
[49] de Toro F, Ortega J, Fernandez J, Diaz A. PSFGA: a parallel genetic algorithm for multiobjective optimization. In: Proceedings of the 10th Euromicro workshop on parallel, distributed and network-based processing, 9–11 January, 2002. Canary Islands, Spain: IEEE Computer Society.
[50] Van Veldhuizen DA, Zydallis JB, Lamont GB. Considerations in engineering parallel multiobjective evolutionary algorithms. IEEE Trans Evol Comput 2003;7(2):144–73.

[51] Wilson LA, Moore MD, Picarazzi JP, Miquel SDS. Parallel genetic algorithm for search and constrained multi-objective optimization. In: Proceedings of the 18th international parallel and distributed processing symposium, 26–30 April, 2004. Santa Fe, NM, USA: IEEE Computer Society; 2004.

[52] Xiong S, 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 August, 2003. Chengdu, China: IEEE; 2003.

[53] Knowles JD, Corne DW. M-PAES: a memetic algorithm for multiobjective optimization. In: Proceedings of the 2000 congress on evolutionary computation, 16–19 July, 2000. La Jolla, CA, USA: IEEE; 2000.

[54] Paquete L, Stutzle T. A two-phase local search for the biobjective traveling salesman problem. In: Evolutionary multi-criterion optimization. Proceedings of the second international conference, EMO 2003, 8–11 April, 2003. Faro, Portugal: Springer; 2003.

[55] Deb K, Goel T. A hybrid multi-objective evolutionary approach to engineering shape design. In: Evolutionary multi-criterion optimization. Proceedings of the first international conference, EMO 2001, 7–9 March, 2001. Zurich, Switzerland: Springer; 2001.

[56] Ishibuchi H, Yoshida T, Murata T. Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. IEEE Trans Evol Comput 2003;7(2): 204–23.

[57] Tan KC, Lee TH, Khor EF. Evolutionary algorithms with dynamic population size and local exploration for multiobjective optimization. IEEE Trans Evol Comput 2001;5(6):565–88.

[58] Marseguerra M, Zio E, Podofillini L. Optimal reliability/availability of uncertain systems via multi-objective genetic algorithms. IEEE Trans Reliab 2004;53(3):424–34.

[59] Martorell S, Villanueva JF, Carlos S, Nebot Y, Sanchez A, Pitarch JL, et al. RAMS + C informed decision-making with application to multi-objective optimization of technical specifications and maintenance using genetic algorithms. Reliab Eng Syst Safety 2005;87(1): 65–75.

[60] Martorell S, Sanchez A, Carlos S, Serradell V. Alternatives and challenges in optimizing industrial safety using genetic algorithms. Reliab Eng Syst Safety 2004;86(1):25–38.

[61] Sasaki M, Gen M. A method of fuzzy multi-objective nonlinear programming with GUB structure by hybrid genetic algorithm. Int J Smart Eng Syst Des 2003;5(4):281–8.

[62] Yang J-E, Hwang M-J, Sung T-Y, Jin Y. Application of genetic algorithm for reliability allocation in nuclear power plants. Reliab Eng Syst Safety 1999;65(3):229–38.

[63] Elegbede C, Adjallah K. Availability allocation to repairable systems with genetic algorithms: a multi-objective formulation. Reliab Eng Syst Safety 2003;82(3):319–30.

[64] Deb K, Jain P, Gupta NK, Maji HK. Multiobjective placement of electronic components using evolutionary algorithms. IEEE Trans Components Packaging Technol 2004;27(3):480–92.

[65] Kumar R, Parida PP, Gupta M. Topological design of communication networks using multiobjective genetic optimization. In: Proceedings of the 2002 world congress on computational intelligence—WCCI'02, 12–17 May, 2002. Honolulu, HI, USA: IEEE; 2002.