Chapter II

Multi-objective Evolutionary Algorithm

The genetic algorithm (GA) has its roots in theory of evolution. It can be described as a stochastic search or optimization technique in which the search mechanisms are based on the Darwinian concept of the survival of the fittest. It was firstly developed by Holland and his colleagues [3] in order to model the mechanics of natural selection. The genetic algorithm became widespread after Goldberg wrote the book [4] which described a concise introduction to a genetic algorithm. Subsequently the method has been widely used in various areas including engineering, scientific, business and medical areas.

In a GA, the problem is translated into the GA framework such that a solution is coded using strings of characters which represent decision variables of the optimization problem. These strings are usually referred to as chromosomes and are made up of a set of characters which are technically called genes. The GA is a parallel search where solutions in one search iteration or generation are transformed by genetic operators in order to obtain better solutions in the next generation. Transformations on solutions or individuals within the population in one generation are done to explore the search space and promote the propagation of fit or desired characteristics within one generation to the next. Fit individuals are expected to emerge from the search where the optimal solution found is represented by the fittest individual obtained.

In this thesis, a genetic algorithm (GA) is used as a multi-objective optimizer. As previously stated in Chapter I, the genetic algorithm for multi-objective optimization is called as "multi-objective evolutionary algorithm (MOEA)". Many MOEAs, GAs for multi-objective optimization had been developed in last decade, for example multi-objective genetic algorithm (MOGA) [40], niched Pareto genetic algorithm (NPGA) [41], non-dominated sorting genetic algorithm (NSGA) [42], strength Pareto evolutionary algorithm (SPEA) [27], fast elitist non-dominated sorting genetic algorithm (NSGA-II) [28], controlled elitist non-dominated sorting genetic algorithm (CNSGA) [29] and improved strength Pareto evolutionary algorithm (SPEA-II) [30]. In addition, a number of purposed MOEAs are the results of the integrations of an additional GA operator into a well-established MOEA. For instance, multi-objective co-operative co-evolutionary genetic algorithm (MOCCGA) [31] and non-dominated sorting co-operative co-evolution genetic algorithm (NSCCGA) [32] integrate the co-operative co-evolution [43] into MOGA [40] and NSGA-II [28] respectively while the multi-objective diversity control oriented genetic algorithm (MODCGA) [33] integrated the diversity control [45], [46], into the MOGA [40].

This thesis proposes 3 MOEAs – the co-operative co-evolution multi-objective algorithm (CCMOA), the improved compressed-objective genetic algorithm (COGA-II), and the co-operative co-evolutionary improved compressed-objective genetic algorithm (CCCOGA-II). The first proposed MOEA, CCMOA, is suitable for multi-objective optimization problems with a few coupling among decision variables. The second MOEA, COGA-II, is presented for optimization problems with three-or-more objectives. The last MOEA, CCCOGA-II, is proposed for three-or-more objectives optimization problems with a few coupling among decision variables. As previously stated in the first chapter, the effectiveness of purposed MOEAs is evaluated by comparing them with the well-established MOEAs which are NSGA-II [28], and SPEA-II [30].

There are seven sections in this chapter. The first section will describe procedures of standard genetic algorithm, which is a single-objective optimizer, and genetic operators. Multi-objective optimization will be discussed in the second topic. After that, the next five sections, the third section to the final section, will describe for five employed MOEAs, NSGA-II, SPEA-II, CCMOA, COGA-II, and CCCOGA-II respectively.

2.1. Standard Genetic Algorithm

The standard genetic algorithm has been extensively explained in [4] and is discussed here to illustrate the basic components and mechanisms of a genetic algorithm. The procedure of the standard genetic algorithm (SGA) in Figure 2.1 can be described as follows.

1) Create an initial population of random chromosomes.

2) Decode the chromosome of every individual in order to obtain solutions of the problem.

3) Based on the solution obtained, calculate the objective value of each individual in the population.

4) Calculate the fitness of each individual, using the obtained objective value.

5) Select a parent population from the current population, using the fitness value as an indicator on how to select individuals from the current population.

6) Perform a transformation on the parent population using genetic operators, crossover and mutation, to obtain the resulting offspring population.

7) Go back to step 2 until a convergence is observed from the solutions found or a fixed number of iterations is reached. Note that one loop from steps 2 to 6 is called one generation of a genetic algorithm run.

By adding the elitism operator, after step 4), then a set of fit individuals might be kept without crossover and mutation, and merged with the newly generated individuals from crossover and mutation in step 6) to form the new population. There are six main genetic operators [4], namely chromosome coding, fitness evaluation, selection, crossover, mutation, and elitism, which are described as the following topics.

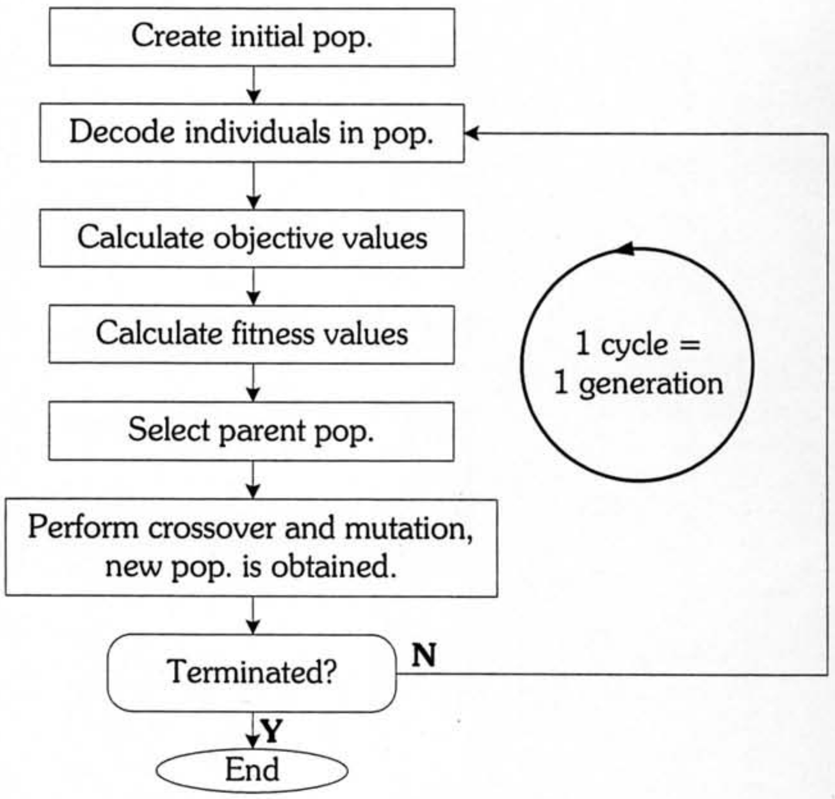
Create initial pop.
Decode individuals in pop.
Calculate objective values
Calculate fitness values
Select parent pop.
Perform crossover and mutation, new pop. is obtained.
Terminated?
N
Y
End
1 cycle = 1 generation

Figure 2.1. SGA procedures.

2.1.1 Chromosome Coding

As previously stated, a solution of an optimization problem is coded as the string of characters or chromosome. There are many chromosome codings such as binary coding, integer coding, and real coding which encode decision variables into strings of binary numbers, integer numbers, and real numbers respectively in which only 2 chromosome codings, binary and real coding, are employed in this thesis.

The binary coding is described as the followings; for example a solution $\mathbf{x}$ is a 3 variables vector $[x_1, x_2, x_3]$, $x_1 \in [-1,1]$, $x_2 \in [0,1]$, $x_3 \in [1,5]$. In this coding, x_1, x_2 and x_3 are coded by a string with the lengths of 4, 5, and 6, respectively. Therefore the representing chromosome of a solution $\mathbf{x}$ is coded as the binary string with the length of $4+5+6 = 15$. For example $\mathbf{x}$ is coded as chromosome

110001011100101, of which x_1, x_2, x_3 are represented by 3 bit partitions in the chromosome as in the following equation.

$$\mathbf{x} = \underbrace{1100}_{x_1}\underbrace{01011}_{x_2}\underbrace{100101}_{x_3} \tag{2.1}$$

Given υ_1, υ_2 and υ_3 are corresponding decoded decimal numbers of binary string partitions of x_1, x_2, and x_3 respectively and are simply evaluated as follows.

$$\upsilon_1 = (1100)_2 = 1(2^3) + 1(2^2) + 0(2^1) + 0(2^0) = 12$$

and $$\upsilon_2 = (01011)_2 = \ldots = 11 \text{ and } \upsilon_3 = (100101)_2 = 37 \tag{2.2}$$

Thereafter x_i can be simply encoded as the following equation.

$$x_i = x_i^l + \upsilon_i\left(x_i^u - x_i^l\right)/\left(2^{L_i} - 1\right) \tag{2.3}$$

where x_i^l, x_i^u and L_i are the lower limit, upper limit and length of corresponding string of x_i. By the above equation, x_1, x_2, and x_3 are decode as 0.6, 0.3548, and 3.3492, respectively.

On the other hand, in real coding the solution $\mathbf{x}$ can be directly coded by the real-coded chromosome with three bits string of which any bit i directly represents for x_i. For the above example, $\mathbf{x}$ is therefore coded by 3-bit chromosome as follow.

$$\mathbf{x} = \underbrace{0.6}_{x_1}\underbrace{0.3548}_{x_2}\underbrace{3.3492}_{x_3} \tag{2.4}$$

After an individual is encoded into the corresponding solution, its objective value is then evaluated from this encoded solution.

2.1.2 Fitness Evaluation

After the objective value is obtained, the fitness, which is a maximization criterion, of each individual is directly evaluated by its objective. The fitness evaluation of single-objective genetic algorithm is quite easy; it depends on the type of single-objective optimization problem – either maximization or minimization problems. For the maximization problem, a fitness of an individual is directly equal to a constant plus its objective. In the other hand, for the minimization problem, the fitness of an individual is equal to a constant minus its objective. The chosen values of the additional constant in both types should guarantee that a fitness value of any individual in a current population is positive.

2.1.3 Selection

The purpose the selection operator is to copy the good solutions and eliminate the bad solutions in the current population for the mating pool. The size of the population (*N*) to be selected is not necessary equal to the size of the mating pool (*P*). Only one type of selection, which is a binary selection, is employed in this thesis. In the binary selection, two solutions are randomly picked from the population. Then, the two solutions are compared each other. The better solution with better fitness is selected into the mating pool. The selection is repeated until the mating pool is fully filled.

2.1.4 Crossover

As mentioned earlier, a genetic algorithm explores the search space by transforming the selected individuals, obtained from the selection process, into new individuals. These transformation processes which are referred to here are crossover and mutation operations. In the transformation process, two individuals from the mating pool are randomly picked out, and referred to as parent individuals. By crossover, the genes of the individuals are passed onto two offspring individuals. The crossover is not always done on all individual, the possibility that a crossover occurs for any selected individual pair is given by the crossover probability.

The crossover procedure is briefly described as follows. Firstly, two parent individuals are randomly selected and removed from the mating pool. After that, a number from 0 to 1 is randomly generated and then compared with a crossover probability. If the random value is less than or equal to the crossover probability, the parent individuals are performed crossover to obtain two resulting offspring individuals, otherwise the offspring individuals are identical to the parent individuals. The crossover process of other parent pair is repeated until the mating pool is empty. After crossover, the number of offspring individuals is equal to the number of individuals in the mating pool.

Two crossovers which are uniform crossover and simulated binary crossover will be presented in this thesis; the first crossover is used for for binary chromosomes, while the last crossover is used for real-coded chromosomes. These two crossovers will be described in the following topics.

1) Uniform Crossover

In this crossover technique, each gene or bit along the whole chromosome of one individual offspring is randomly picked from the corresponding gene location on the parents' chromosomes. Usually, the possibility of a gene being chosen from one parent individual is equal to that of a gene being chosen from the other parent individual. Once a gene is selected for one offspring individual, the remaining gene for the same locus will be given to the other offspring individual. A schematic diagram describing uniform crossover is given by an example of the crossover in Figure 2.2.

$$\begin{array}{ll} \textit{parent 1} & 1\,1\,1\,1\,1\,1\,1\,1 \\ \textit{parent 2} & 0\,0\,0\,0\,0\,0\,0\,0 \end{array} \xrightarrow{\textbf{crossover}} \begin{array}{ll} \textit{offspring 1} & 1\,0\,1\,1\,0\,1\,0\,1 \\ \textit{offspring 2} & 0\,1\,0\,0\,1\,0\,1\,0 \end{array}$$

Figure 2.2. An example of uniform crossover.

2) Simulated Binary Crossover

Deb [47] developed the simulated binary crossover (SBX) which adapts the one-point crossover on binary strings for real-coded chromosomes. In this crossover, a probability distribution is used around parent solutions to create two offspring solutions.

As an example for one-bit real-coded chromosome without lower and upper bound, offspring individuals y_i^1 and y_i^2 are computed from parent individuals x_i^1 and x_i^2 where $x_i^1 \leq x_i^2$. They are created by the use of a probability distribution around the parent individuals. The absolute difference in offspring values to that of the parent is a spread factor β_i and given by the following equation.

$$\beta_i = \left| \frac{y_i^2 - y_i^1}{x_i^2 - x_i^1} \right| \tag{2.5}$$

The probability distribution used to created offspring individuals is derived to have similar search power as that in a single-point crossover in the binary-coded chromosomes. The probability distribution is given by the equation (2.6) and its profile is shown in the Figure 2.3.

$$P(\beta_i) = \begin{cases} 0.5(\eta_c + 1)(\beta_i)^{\eta_c}, & \text{if } \beta_i \leq 1 \\ 0.5(\eta_c + 1)/(\beta_i)^{\eta_c+2}, & \text{otherwise} \end{cases} \tag{2.6}$$

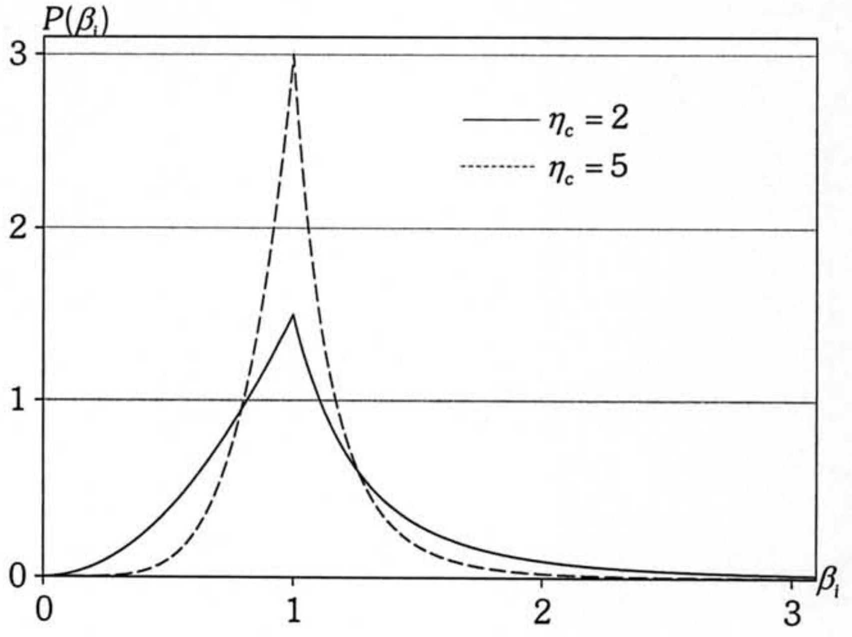

Figure 2.3. Probability distribution profile.

where η_c is the distribution index; which can be any non-negative real number. A value of η_c has an effect on the distance between parent and offspring individuals which will be described later. From the equation (2.6), $\int_0^\infty P(\beta_i)=1$, $\int_0^1 P(\beta_i)=0.5$, and $\int_1^\infty P(\beta_i)=0.5$. In each crossover, a random number u from 0 to 1 is created, then the parameter β_{u_i} is found so that the area under curve $P(\beta_i)$ from 0 to β_{u_i} is equal to u or $\int_0^{\beta_{u_i}} P(\beta_i)=u$. The β_{u_i} can be evaluated from the integral which is given by the following equation.

$$\beta_{u_i}=\begin{cases}(2u_i)^{1/(\eta_c+1)}, & \text{if } u_i \le 0.5\\ \left(1/2(1-u_i)\right)^{1/(\eta_c+1)}, & \text{otherwise}\end{cases} \tag{2.7}$$

After β_{u_i} is obtained, the offspring individuals y_i^1 and y_i^2 can be computed by the equation (2.8) and (2.9), respectively.

$$y_i^1=0.5\left[\left(1+\beta_{u_i}\right)x_i^1+\left(1-\beta_{u_i}\right)x_i^2\right] \tag{2.8}$$

$$y_i^2=0.5\left[\left(1-\beta_{u_i}\right)x_i^1+\left(1+\beta_{u_i}\right)x_i^2\right] \tag{2.9}$$

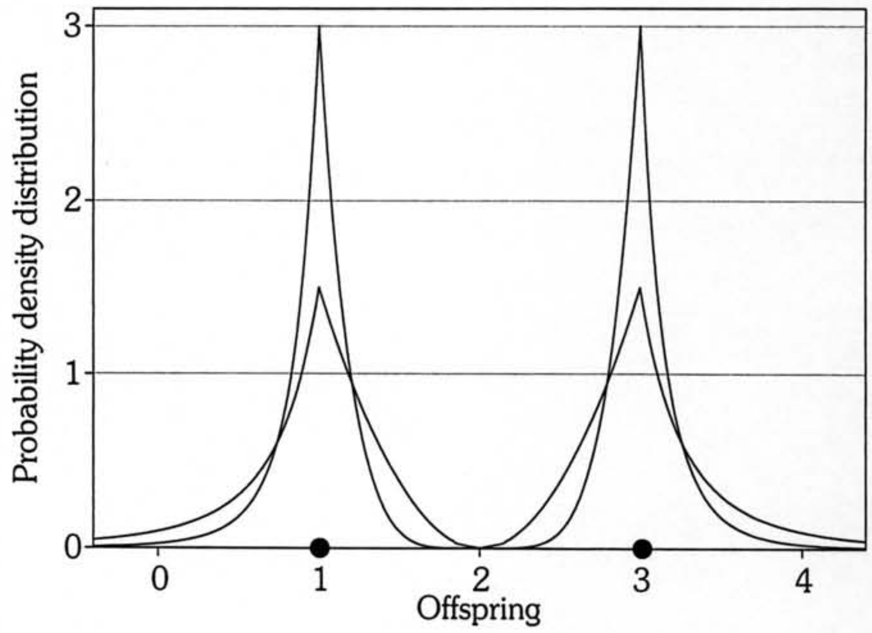

Figure 2.4. Probability distribution for creating offspring individuals.

From the equations (2.8) and (2.9), it can be noted that $\left|(y_i^2 - y_i^1)/(x_i^2 - x_i^1)\right| = \beta_{u_i}$ as in the equation (2.5). Figure 2.4 shows a probability density distribution with η_c = 2 and 5 for creating offspring individuals from two parent individuals x_i^1 = 1.0 and x_i^2 = 3.0. A large value of η_c gives a higher probability for creating near parent individuals and a small value of η_c allows distant solutions to be selected as children individuals.

For lower and upper boundaries (x^l and x^u) are specified, two parameter $\beta_{u_i}^1$ and $\beta_{u_i}^2$ are used to calculate y_i^1 and y_i^2, respectively; they are evaluated by equations (2.10) and (2.11).

$$\beta_{u_i}^1 = \begin{cases} (\alpha_1 u)^{1/(\eta_c+1)}, & u \le \dfrac{1}{\alpha_1} \\ (1/(2-\alpha_1 u))^{1/(\eta_c+1)}, & \text{otherwise} \end{cases} \tag{2.10}$$

$$\beta_{u_i}^2 = \begin{cases} (\alpha_2 u)^{1/(\eta_c+1)}, & u \le \dfrac{1}{\alpha_2} \\ (1/(2-\alpha_2 u))^{1/(\eta_c+1)}, & \text{otherwise} \end{cases} \tag{2.11}$$

where the parameters α_1 and α_2 are evaluated as the equations (2.12) and (2.13), respectively.

$$\alpha_1 = 2 - \beta_1^{-(\eta_c-1)},\ \beta_1 = 1 + 2\left(x_1 - x^l\right)/\left(x_2 - x_1\right) \tag{2.12}$$

$$\alpha_2 = 2 - \beta_2^{-(\eta_c-1)},\ \beta_2 = 1 + 2\left(x^u - x_2\right)/\left(x_2 - x_1\right) \tag{2.13}$$

Thus, the offspring individuals y_i^1 and y_i^2 can be calculated by the equation (2.14) and (2.15), respectively.

$$y_i^1 = 0.5\left\{\left(x_i^1 + x_i^2\right) + \beta_{u_i}^1\left(x_i^2 - x_i^1\right)\right\} \tag{2.14}$$

$$y_i^2 = 0.5\left\{\left(x_i^1 + x_i^2\right) + \beta_{u_i}^2\left(x_i^2 - x_i^1\right)\right\} \tag{2.15}$$

The above description is the derivation of the SBX crossover for single-bit chromosomes. For multiple-bit chromosomes, similar to a uniform crossover in binary-code chromosome, two parent genes x_i^1 and x_i^2 at a same location are crossed by a probability 0.5. If the crossover is permitted, the parents are crossed to obtain offspring genes y_i^1 and y_i^2 as single-bit chromosomes, y_i^1 and y_i^2 are randomly switched to produce corresponding genes of offspring chromosomes.

2.1.5 Mutation

A mutation is used to transform an offspring individual into a new individual. It is used to maintain the diversity of individuals in a population preventing the premature convergence of solution. A crossover operation creates new individuals which are certain distances away in search space from their parent individuals. Then, a mutation operation can be thought as a small perturbation on the chromosome of an individual. In other words, mutation leads to a search at a neighboring point from the original search point of the offspring as dictated by the structure of the chromosome.

As in crossover, the mutation of an offspring individual is also governed by the mutation probability. In the mutation, a corresponding number, whose value is from 0 to 1, of a gene in an offspring individual is randomly generated. If the number is less than or equal to the mutation probability, the gene of the offspring is mutated, otherwise it remains unchanged. There are two mutations for binary-coded and real-coded chromosomes in this thesis which are presented in two following topics.

1) Bit-Flipped Mutation

For a binary chromosome, a mutation can be achieved by reversing the allele value of a gene. Hence, this kind of mutation is generally referred to as bit-flipped mutation. A schematic diagram of bit-flipped mutation is illustrated in Figure 2.5.

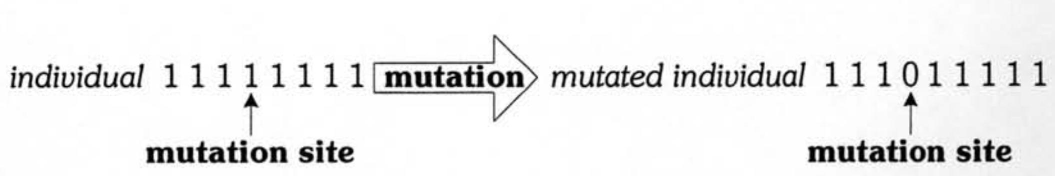
individual 1 1 1 1 1 1 1 1
mutation
mutated individual 1 1 1 0 1 1 1 1
mutation site
mutation site

Figure 2.5. Bit-flipped mutation.

In Figure 2.5, the mutation site is located at the locus 4 on the chromosome. The gene at this locus changes from 1 to 0. Similar to the crossover operation, mutation will not occur on every gene on the chromosome as the possibility of a mutation for one particular gene in the chromosome is governed by the mutation probability.

Generally, mutation probability is set to a value in the range of 0 to 0.1; usually, the mutation probability also varies with the length of a coded chromosome. At this point, it can be clearly seen that with the value of the crossover probability being close to 1 while the value of the mutation probability being close to 0, the main driving force behind the exploration of search space in a genetic algorithm from the crossover operation. It is also important to stress that crossover and mutation operations are done in the hope that a recombination (crossover) of fit genes, exhibited in the selected individuals, and an introduction of new genes to the population (mutation) would lead to the creation of offspring individuals which are fitter than their parent individuals.

2) Variable-Wise Polynomial Mutation

For real-coded chromosomes, a polynomial probability distribution is used to create a gene y in a mutated individual the vicinity of a corresponding gene x in an offspring individual [48].

In the case of no lower and upper boundaries, firstly a random number u whose value is from 0 to 1 is generated. The parameter $\bar{\delta}$ is calculated as the following equation:

$$\bar{\delta} = \begin{cases} (2u)^{1/(\eta_m+1)} - 1, & \text{if} \quad u \le 0.5, \\ 1 - \left[2(1-u)\right]^{1/(\eta_m+1)}, & \text{otherwise} \end{cases} \tag{2.16}$$

where η_m is the distribution index for mutation and takes any non-negative value. The gene y is evaluated as the following equation:

$$y = x + \bar{\delta}\Delta_{\max} \tag{2.17}$$

where $\Delta_{\max}$ is the maximum perturbation allowed in the considering gene in the offspring individual.

For the gene whose lower and upper boundaries (x^l and x^u) are specified, the equation of parameter $\bar{\delta}$ is changed as following equation:

$$\bar{\delta} = \begin{cases} \left[2u + (1-2u)(1-\delta)^{1/(\eta_m+1)}\right] - 1, & \text{, if} \quad u \le 0.5 \\ 1 - \left[2(1-u) + 2(u-0.5)(1-\delta)^{\eta_m+1}\right]^{1/(\eta_m+1)} & \text{,otherwise} \end{cases} \tag{2.18}$$

where $\delta = \min[(x - x^l), (x^u - x)]/(x^u - x^l)$ and $\Delta_{\max} = x^u - x^l$. The defined value of δ guarantees that no mutated genes are outside the range $[x^l, x^u]$. In the equations (2.17) and (2.18), the mutated gene y is in the negative and positive sides of the corresponding gene x for $u < 0.5$ and $u > 0.5$, respectively. In addition, the value of normalized perturbation $(y-x)/(x^u - x^l)$ or $\bar{\delta}$ is the same order as that of $1/\eta_m$ [9]. This implies that, for example, in order to get mutation effect of 5% perturbation in mutated genes, η_m should be set to 20.

2.1.6 Elitism

Elitism can generally be described as a special way which is incorporated into a genetic algorithm in order to promote the survival of the best individual found. The main reason behind the use of elitism is the fact that there is always a possibility that a crossover or mutation operation might eliminate the best individual found within each generation, subsequently, the best individual in each generation might be worst than that of the previous generation. In order to guarantee that the best solution found from each generation is not lost though the search propagation. The most commonly used elitism is to pass the first *n* number of best individuals from the current generation to the next generation without crossover and mutation.

2.2. Multi-objective Optimization

This section will briefly describe a multi-objective optimization of which the first and the second subsection will present a general form of a multi-objective optimization problem and two approaches to multi-objective optimization respectively. The subsections are as the follows.

2.2.1. Multi-objective Optimization Problem

A multi-objective optimization problem (MOOP) has a number of objective functions which are to be minimized or maximized. In the following, the multi-objective optimization problem (MOOP) is stated in its general form [5] as the following equation.

$$\begin{gathered}\text{Minimize/Maximize } f_m(\mathbf{x}),\ m=1,2,\ldots,M \\ \text{subject to } g_j(\mathbf{x})=0,\ j=1,2,..,J \\ h_k(\mathbf{x}) \geq 0,\ k=1,2,\ldots,K \\ x_i^{(L)} \leq x_i \leq x_i^{(U)},\ i=1,2,\ldots,n. \end{gathered} \tag{2.19}$$

A solution $\mathbf{x}$ is a vector of n decision variables: $\mathbf{x} = \{x_1,x_2,..,x_n\}$. There are M objective functions $\mathbf{f} = \{f_1(\mathbf{x}), f_2(\mathbf{x}),\ldots,f_M(\mathbf{x})\}$; each objective function $f_m(\mathbf{x})$ can be either minimized or maximized. For two sets of J equality and K inequality constraints, $g_j(\mathbf{x})$ and $h_k(\mathbf{x})$ are equality and inequality constraint functions respectively. Although the inequality function $h_k(\mathbf{x})$ which is treated as "greater than or equal to", it can represent for constraint function of "less than or equal to" by multiplying the function with -1. The last set of constraint is variable bounds, restricting each decision variable x_i to take a value within a lower $x_i^{(L)}$

and an upper $x_i^{(U)}$ bound. These bounds constitute a decision variable space S. This thesis will present two approaches used to solve the multi-objective optimization problem which will be described in the following topic.

2.2.2. Two Approaches to Multi-Objective Optimization

These two approaches, which are used as multi-objective optimizer, are an aggregating approach or a weighted-sum approach, and a Pareto-based approach. Their procedures are described in Figure 2.6 and Figure 2.7 (the figures are obtained from [5]), respectively. In the aggregating approach, the classical multi-objective optimization approach, all optimization objectives are weighted and added together in order to form a single objective. Thereafter a single-objective optimizer is used to solve this newly formed single-objective problem in the usual way to obtain one optimum solution. Unless a reliable and accurate preference vector is available, the optimal solution obtained by such approaches is highly subjective to the particular user. Although only one best compromised solution is obtainable after this form of search strategy, this technique has proved to be popular due to its simplicity. Since it is not possible to obtain a single solution in which all optimization objectives are optimized simultaneously from a multi-objective optimization problem, the true solutions to the problem will be non-dominated solutions or true Pareto optimal solutions. This can be solved by the Pareto-based approach, the second approach, employs the Pareto domination in the following Definition 2.1 to compare solutions of a MOOP. In this approach, multiple trade-off solutions are found, then, by higher level qualitative consideration, one suitable solution is selected for the problem. This thesis will use the Pareto-based approach to solve a multi-objective optimization problem in order to obtain multiple trade-off solutions. However, the solution selection from multiple trade-off solutions as in Figure 2.7 is very particular, this thesis will not choose one solution for an employed problem.

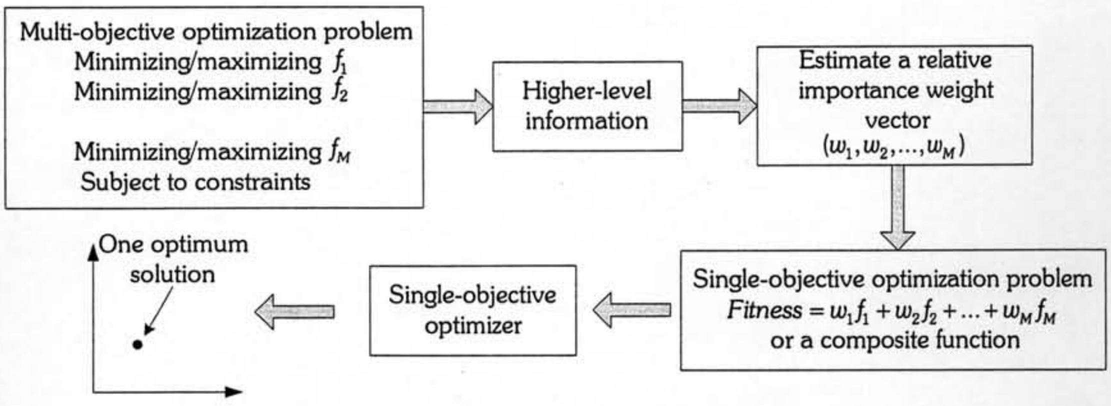
Multi-objective optimization problem
Minimizing/maximizing f_1
Minimizing/maximizing f_2
Minimizing/maximizing f_M
Subject to constraints
Higher-level information
Estimate a relative importance weight vector $(w_1, w_2, \ldots, w_M)$
Single-objective optimization problem
$Fitness = w_1f_1 + w_2f_2 + \ldots + w_Mf_M$
or a composite function
Single-objective optimizer
One optimum solution

Figure 2.6 Aggregating approach.

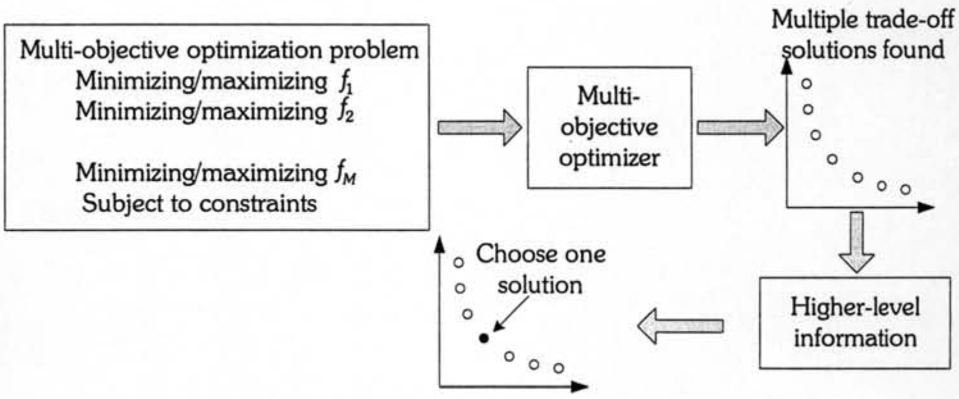
Multi-objective optimization problem
Minimizing/maximizing f_1
Minimizing/maximizing f_2
Minimizing/maximizing f_M
Subject to constraints
Multi-objective optimizer
Multiple trade-off solutions found
Higher-level information
Choose one solution

Figure 2.7 Pareto-based approach.

Definition 2.1 Pareto domination

A solution x dominates the other solution y or $x \prec y$ if the two conditions are satisfied:

a) For any objective, the solution x is not worse than the solution y.

b) At least one objective, solution x is better than solution y.

Fonseca and Fleming [40] have also incorporated the use of a goal attainment method for domination. The modified domination scheme can be described as follows. Assuming a multi-objective optimization problem, consider two M-dimensional objective vectors, $\mathbf{x} = \{x_1, x_2, x_3, \ldots, x_M\}$ and $\mathbf{y} = \{y_1, y_2, y_3, \ldots, y_M\}$ and the goal vector $\mathbf{g} = \{g_1, g_2, g_3, \ldots, g_M\}$ The notation $x_i \trianglelefteq g_i$ means x_i meet a goal g_i or x_i is better than or equal to g_i, in the contrast, $x_i \ntrianglelefteq g_i$ means x_i

does meet the goal g_i. If k is the number of components that $\mathbf{x}$ does not meet the goal, then, q-k components of $\mathbf{x}$ meet the goal, this can be written as the following equation.

$$\exists k = 1,\ldots,q-1 : \forall i = 1,\ldots,k, \forall j = k+1,\ldots,q, (x_i \ntrianglelefteq g_i) \wedge (x_j \trianglelefteq g_j) \tag{2.20}$$

which assumes a convenient permutation of the objectives. The objective vector $\mathbf{u}$ is said to be dominating the objective vector $\mathbf{v}$ if one of three conditions, each of the conditions is presented by each of three following equations, is satisfied.

$$(\mathbf{u}_{(1,\ldots,k)} \prec \mathbf{v}_{(1,\ldots,k)}) \vee \{(\mathbf{u}_{(1,\ldots,k)} = \mathbf{v}_{(1,\ldots,k)}) \wedge$$
$$\left[(\mathbf{x}_{(k+1,\ldots,q)} \prec \mathbf{y}_{(k+1,\ldots,q)}) \vee \sim(\mathbf{y}_{(k+1,\ldots,q)} \trianglelefteq \mathbf{g}_{(k+1,\ldots,q)})\right]\},\ 1 \le k \le \mathrm{M}\text{-}1, \tag{2.21}$$

$$\mathbf{u} \prec \mathbf{v},\ \text{for } k = \mathrm{M}, \tag{2.22}$$

$$(\mathbf{x} \prec \mathbf{y}) \vee \sim(\mathbf{x} \trianglelefteq \mathbf{g}),\ \text{for } k = 0. \tag{2.23}$$

An early attempt to use a genetic algorithm to solve a multi-objective optimization problem includes the application of a vector evaluated genetic algorithm (VEGA) [49], which employs the aggregating approach, as suggested by Schaffer, on benchmark problems. However, the solutions produced by a VEGA tend to be solutions in which some of the objective values are at their extreme. Solutions with objective values which are not extreme tend to be left out especially in the non-convex multi-objective optimization problems [5]. This is a weak point of the aggregation approach, in the other hand, a multi-objective evolutionary algorithm (MOEA) based on the Pareto approach does not have such weak point, then, many MOEAs based on the approach are purposed. Original Pareto-based MOEAs are non-elitist MOEAs such as multi-objective genetic algorithm (MOGA) [40], niched Pareto genetic algorithm (NPGA) [41], non-dominated sorting genetic algorithm (NSGA) [42] while a modern Pareto-based MOEA is an elitist MOEA such as strength Pareto evolutionary algorithm (SPEA) [27], non-dominated sorting genetic algorithm II (NSGA-II) [28], Pareto-archive evolution strategy (PAES) [50], and improved strength Pareto evolutionary algorithm (SPEA-II) [30].

In this thesis 5 MOEAs are used as multi-objective optimizers, the first two MOEAs are the well-established elitist MOEAs – the non-dominated sorting genetic algorithm II (NSGA-II) and the improved strength Pareto evolutionary algorithm (SPEA-II), the other multi-objective optimizers are the new purposed MOEAs, the co-operative co-evolutionary multi-objective algorithm (CCMOA), the improved compressed-objective genetic algorithm (COGA-II), and the co-operative co-evolutionary improved compressed-objective genetic algorithm (CCCOGA-II). The employed MOEAs will be discussed in the following sections.

2.3. Non-dominated Sorting Genetic Algorithm II (NSGA-II)

A fas elitist non-dominated sorting genetic algorithm (NSGA-II), which is purposed by Deb [28], is quit different from the original algorithm, a non-dominated sorting genetic algorithm (NSGA). Only its ranking is the same as that of NSGA, in the other hand it does not require a niching parameter (σ_{share}), which is a weak point of a multi-objective evolutionary algorithm (MOEA) use this parameter [51], as in the NSGA. The NSGA-II proposed a crowding distance assignment to maintain diversity of solutions. Its procedure is as follows.

2.3.1. NSGA-II main algorithm

1) Create random initial population P_0 of size N and set $t = 0$.

2) Assign rank for an individual in the population P_t.

3) Evaluate a crowding distance of an individual of each rank in P_t. The crowding distance of an individual indicates diversity of the individual.

4) Select parent population (A_t) from the current population P_t by binary selection which judges with ranks and crowding distances. For each time of binary selection, two solutions are randomly selected, if the ranks of the

solutions are not equal, the solution with higher rank will be chosen, otherwise, their ranks are equal, for diversity of parent population (A_t), the solution with more crowding distance is chosen.

5) Apply crossover and mutation to A_t to form a offspring population Q_t of size N.

6) Merge the current population P_t and offspring population Q_t to form a merged population R_t of size $2N$.

7) Assign rank for an individual and evaluate crowding distance of each individual in the merged population R_t.

8) Obtain a new population P_{t+1} of size N from R_t of size $2N$ by NSGA-II truncation.

9) Check a termination condition. If the condition is satisfied, the algorithm is stop, the non-dominated solutions of final population $Pt+1$ is the output of the algorithm, otherwise increase t to $t+1$ and go back to 2).

Three particular processes, rank assignment, crowding distance evaluation and NSGA-II truncation, of the NSGA-II will be described in the following topics.

2.3.2. NSGA-II Rank Assignment

As previously stated, a rank assignment of the non-dominated sorting genetic algorithm II (NSGA-II) is the same as that of the original algorithm, the non-dominated sorting genetic algorithm (NSGA) [42]. In the original algorithm, for a solution set P of size N, non-dominated solutions have the highest rank (rank = 1), they are then removed from the set. Subsequently, the non-dominated solutions of the remaining set are assigned the next rank or rank 2, and are removed from the current solution set again. The ranking process is repeated again until the remaining solution set is empty. The NSGA-II presents the fast non-dominated sorting for its ranking as the following pseudo-code in Figure 2.8 (which is obtained from [28]).

```
Fast-non-dominated-sorting ($P$)
For each $p \in P$
  $S_p = \phi$
  $F_1 = \phi$
  $n_p = 0$
    For each $q \in P$
      If ($p \prec q$) then                If $p$ dominates $q$
        $S_p = S_p \cup \{q\}$              Add $q$ to the set of solutions dominated by $p$
      Else if ($q \prec p$) then           $p$ belongs to the first rank
        $n_p = n_p+1$
    If $n_p = 0$ then
      rank$_p$ = 1
      $F_1 = F_1 \cup \{p\}$
$i$ =1                                     Initialize the front counter
while $F_i \neq \phi$
  Q = $\phi$                               Used to store the members of the next rank
  For each $p \in Fi$
    For each q $\in Sp$
      $n_q = n_q$-1
      if $n_q$ = 0 then                   q belongs to the next rank
        rank$_q = i+1$
        Q = Q $\cup \{q\}$
  $i = i + 1$
  $F_i$ = Q
```

Figure 2.8 Pseudo-code of fast-non-dominated sorting.

2.3.3. Crowding Distance Evaluation

NSGA-II purpose a crowding distance indicates diversity of a solution in a population. The sharing of the original NSGA is replaced with a crowd comparison method, which sort the population according to the crowding distance of objective values. The crowd comparison method does not require any user-defined parameter such as sharing factor for maintaining diversity among population members. The crowding distance evaluation requires sorting the population according to each objective function. After that a crowding distance of a solution corresponding to each objective function is evaluated as the followings.

The corresponding crowding distance of a boundary solution is equal to infinity, otherwise, that of an interior solution is equal to the absolute normalized difference in the function values of two adjacent solutions. Figure 2.9 shows the difference of two adjacent solutions of a non-extreme solution i in objective j (d_i^j), where $f_{\min}^j$ and $f_{\max}^j$ are the minimum and maximum values of objective function j. The corresponding crowding distance of the interior solution i for objective j (cd_i^j) is given by the equation (2.24).

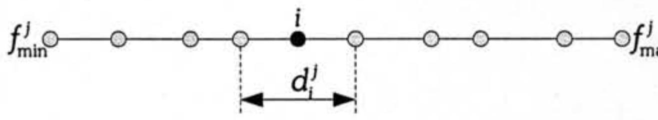

Figure 2.9 Corresponding crowding distance to objective j.

$$cd_i^j = \frac{d_i^j}{\left(f_{\max}^j - f_{\min}^j\right)} \tag{2.24}$$

After the corresponding crowding distance of an individual to each objective is obtained, the overall crowding distance of the individual is computed as the sum of its corresponding crowding distance for each objective function. The crowding distance of a solution i (cd_i) for an M-objective optimization problem is given by the following equation.

$$cd_i = \sum_{j=1}^{M} cd_i^j \tag{2.25}$$

The crowding distance is used not only in the binary selection but also in the truncation. The truncation of NSGA-II will be described as the following topic.

2.3.4. NSGA-II Truncation

After the merging of the current population P_t and offspring population Q_t, the new population P_{t+1} of size N is picked out from the merged population R_t of size $2N$ by NSGA-II truncation. At first, members of the merged population R_t is sorted according to their ranks from the highest rank (rank 1) to the lowest rank, afterward, P_{t+1} is formed. If the number of members in rank 1 is less than N, all members in the rank are put into the P_{t+1}, then, members in a next rank, rank 2, are considered to be chosen into P_{t+1}, the consideration for the rank is similar to that for the rank 1. If the number of members of rank 1 and rank 2 is less than N, all members in the rank are chosen, then, the members in the next rank are considered. The process is repeated until in the rank i such that the accumulated number of members of rank 1 to rank i is more than or equal to N. If the number is equal to N, the process is finished; otherwise if the number is more than N, the first members with the most crowding distances are fully filled P_{t+1}. The procedure of NSGA-II truncation is also described in Figure 2.10.

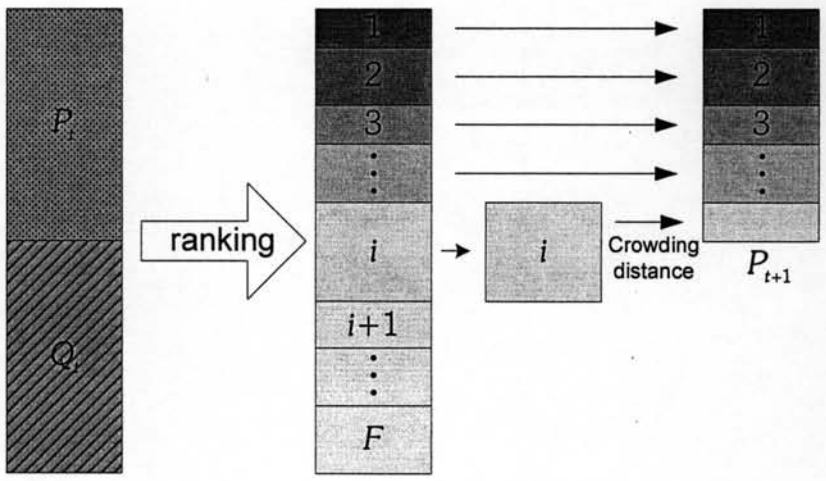

Figure 2.10 NSGA-II truncation.

Although the crowding distance, which is employed in both the binary selection and truncation, is quite quick to evaluated, it may not be the good diversity indicator of solutions in a three-or-more objectives optimization problem, therefore NSGA-II may constitute non-diversity solutions to this problem [37].

2.4. Improved Strength Pareto Evolutionary Algorithm

An improved strength Pareto evolutionary algorithm (SPEA-II) [30] is an improved version of the original algorithm, a strength Pareto evolutionary algorithm (SPEA) [27], which incorporates in contrast to its predecessor a finegrained fitness assignment strategy, a density estimation technique, and an enhanced archive truncation method. Given N and $\bar{N}$ are population size and archive size, unlike the original algorithm SPEA, the number of individuals in archive of SPEA-II is constant in every generation, the overall algorithm of SPEA-II is as the following topic.

2.4.1. SPEA-II Main Algorithm

1) Create random initial population P_0 of size N and create the empty archive (external set) $\bar{P}_0$ set $t = 0$.

2) Merge current population P_t and current archive $\bar{P}_t$ to form merged population R_t and then calculate fitness values of individuals in R_t.

3) Put all non-dominated solutions in R_t to the new archive $\bar{P}_{t+1}$. If size of $\bar{P}_{t+1}$ is more than archive size $\bar{N}$ then reduce $\bar{P}_{t+1}$ by means of the truncation operator, if size of $\bar{P}_{t+1}$ is equal to $\bar{N}$ stop and go to the next step, otherwise, if size of $\bar{P}_{t+1}$ is less than $\bar{N}$ then fill $\bar{P}_{t+1}$ with the best $\bar{N} - |\bar{P}_{t+1}|$ dominated individuals in R_t.

4) Check a termination condition. If the condition is satisfied, the algorithm is stop, the non-dominated solutions of final archive $\bar{P}_{t+1}$ is the output of the algorithm, otherwise go to the next step.

5) Perform binary selection tournament selection with replacement on the current archive $\bar{P}_{t+1}$ in order to fill the mating pool. In SPEA-II, only fitness values are consider in the binary selection.

6) Apply crossover and mutation operators to the mating pool and set P_{t+1} to the resulting population. Increase the generation counter by one ($t \leftarrow t + 1$), go back to 2).

The fitness assignment and archive truncation will be described in detail as follow.

2.4.2. SPEA-II Fitness Assignment

At first, an individual i in merged population R_t is assigned its strength S_i as the number of solutions in R_t that it dominates. S_i is described as the following equation.

$$S_i = \left|\left\{ j \middle| j \in R_t \wedge i \prec j \right\}\right| \tag{2.26}$$

After strengths of all individuals in R_t are obtained, a raw fitness value of an individual i or RF_i is equal to the sum of strengths of individuals dominate it. RF_i is evaluated as the following equation.

$$RF_i = \sum_{j \in R_t \ \wedge j \prec i} S_j \tag{2.27}$$

It is important to note that fitness value is to be minimized as the original algorithm SPEA-II, while a raw fitness of a non-dominated individual is equal to zero. Although the raw fitness assignment provides a reasonable measurement based on the concept of Pareto domination, it may be fail when most individuals do not dominate each other. Therefore, a diversity value of an individual is incorporated to discriminate individuals with identical raw fitness values. The diversity value of an individual is evaluated from the density estimation technique, which is an adaptation of the k-th nearest neighbor method [52], where the density of a point i is inverse variation of its k-th nearest distance or d_i^k. Therefore , in SPEA-II, the density of an individual i or D_i is simply taken to the inverse of d_i^k, where k is equal to square root of the data size [52], then k = $\sqrt{N+\bar{N}}$. An individual with large density means there are many neighboring point near it. It can be noted that if the individual is selected, it contributes a little diversity to the mating pool. This implies that a density of an individual i or D_i is to be minimized as the raw fitness RF_i, its fitness or F_i is then equal to the sum of its raw fitness RF_i and its density D_i, hence the fitness F_i is also to be minimized. In addition, the evaluations of D_i and F_i can be described by the following equations (2.28) and (2.29), respectively.

$$D_i = \frac{1}{d_i^k + 2} \tag{2.28}$$

$$F_i = RF_i + D_i \tag{2.29}$$

In the equation (2.28) for D_i, the denominator is d_i^k plus 2, two is added to ensure that the value of D_i is more than zero and less than one, subsequently, it guarantees that the fitness of individual with smaller (better) raw fitness is also lower than that of individual with larger (worst) raw fitness. In addition, the fitness of a non-dominated individual is less than 1, while the fitness of dominated individual is more than or equal 1.

2.4.3. SPEA-II Archive Truncation

Although [30] stated that the archive truncation of SPEA-II differs from that of its predecessor SPEA, clustering selection, in the respect that it can prevent lose of extreme individuals which may happen by the clustering selection in SPEA. By detailed consideration, it prevents this lose only in a two objectives optimization problem, in the other hand, an extreme individual may be removed from the archive for a three-or-more objectives optimization problem. After non-dominated individuals of size more than the archive size $\bar{N}$ in merged population R_t are put into the archive $\bar{P}_{t+1}$, the archive truncation will iteratively removes individuals from the archive $\bar{P}_{t+1}$ until $|\bar{P}_{t+1}| = \bar{N}$. For each iterative removal, an individual i is chosen to be removed from $\bar{P}_{t+1}$ if for all $j \in \bar{P}_{t+1}$, $i \leq_d j$. The notation $i \leq_d j$ is described by the following equation.

$$\begin{gathered} \forall k \in [1, |\bar{P}_{t+1}| - 1] : d_i^k = d_j^k \ \vee \\ \exists k, k \in [1, |\bar{P}_{t+1}| - 1] : \left[\left(\forall l \in [1, k-1] : d_i^l = d_j^l \right) \wedge d_i^k < d_j^k \right] \end{gathered} \tag{2.30}$$

where d_i^k denotes the k^{th} nearest distance from i to its neighbor in current $\bar{P}_{t+1}$. The first condition in the above equation means that the objective vectors of i is the same as that of j. While the other condition can be described as follows. The individual which has the minimum distance to another individual is chosen at each stage; if there are several individuals with minimum distance the tie is broken by considering the second nearest distance and so on. Figure 2.11 and Figure 2.12 show examples of SPEA-II archive truncation for optimization problems with two and three objectives, respectively. In both figures, there are 8 non-dominated solutions on the left graphs, while a size of archive $\bar{N} = 5$, then 8-5 = 3 solutions which are displayed by the black points on the right graphs are iteratively removed, the first, second, third removed solutions are marked by 1, 2, and 3, respectively.

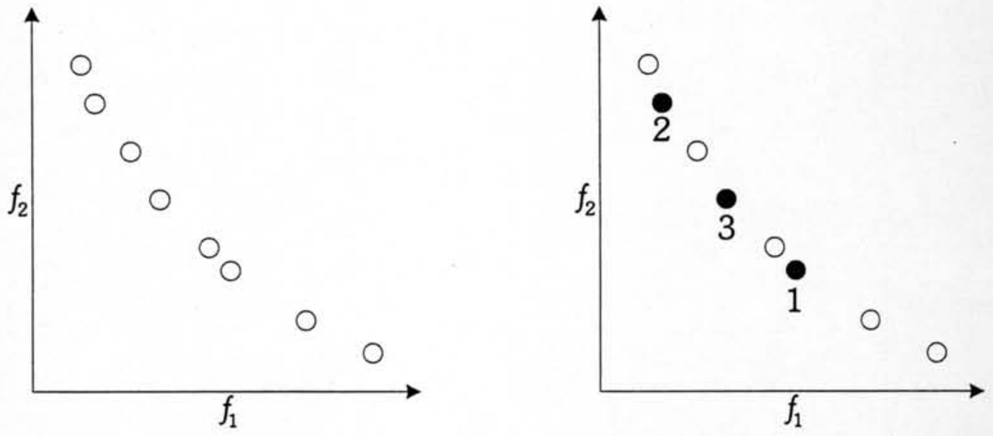

Figure 2.11 SPEA-II archive truncation in a two objectives optimization problem.

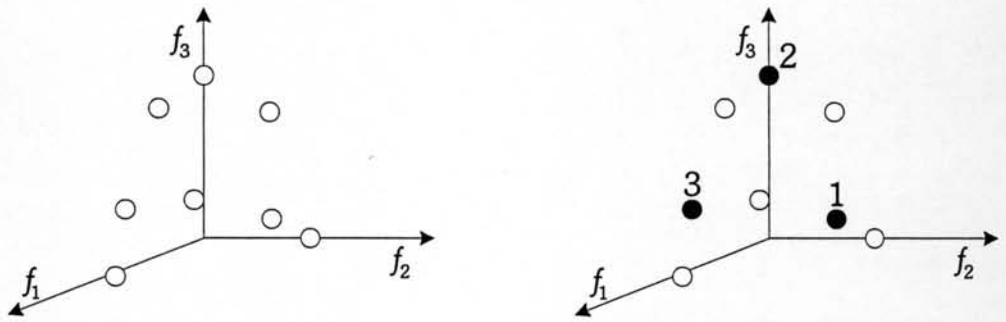

Figure 2.12 SPEA-II archive truncation in a three objectives optimization problem.

Figure 2.11 shows that all extreme individuals are maintain in the archive, however Figure 2.12 shows that one extreme solution, the second removed solution, is removed from the archive. In a two objectives optimization problem, the Pareto front of non-dominated solutions represents by line. Subsequently, if an extreme solution x is compare with its nearest solution y in the current archive, by the conditions in (2.30), $y \leq_d x$, this shows that the extreme solution x is not removed from the current archive in any iterative removal. However, in a three-or-more objectives optimization problem, mostly the Pareto front of non-dominated solutions does not represent by line or curve. It cannot guarantee that $y \leq_d x$, therefore the extreme solution x is probably removed from the current archive in one iterative removal.

2.5. Co-operative Co-evolutionary Multi-Objective Algorithm

A co-operative co-evolutionary multi-objective algorithm (CCMOA) employs the idea of a co-operative co-evolutionary genetic algorithm (CCGA) [43] into multi-objective optimization. The CCMOA is quite different from the integration of CCGA into multi-objective evolutionary algorithms (MOEAs) [31], [32], [35] and others MOEAs that employ co-operative co-evolution such as distributed cooperative coevolutionary algorithm (DCCEA) [44]. For MOEAs employ co-operative co-evolution, a solution is divided into several components or species, each species has a sub-population that represents for the corresponding component of solutions. All species or sub-populations are independently evolved together with the update of an external best solution set. In the other hand, for the CCMOA, a solution is still divided into several components or species; however, the CCMOA does not have several sub-populations, it has only one population as that of traditional MOEAs. The crossover and mutation of CCMOA is quite different from that of the traditional MOEAs, they applied on only some components or species. The main procedure, solution truncation, and crossover and mutation of CCMOA will be described in the following topics.

2.5.1. Main Algorithm

1) Generate an initial population P_0 and an empty archive A_0. Initialize the generation counter ($t = 0$).

2) Merge the population P_t and the archival individuals in A_t together to form the merged population R_t. Then assign ranks to individuals in R_t, rank of an individual i or rank$_i$ is equal to one plus number of its dominators which is given by the following equation.

$$\text{rank}_i = 1 + \left|\{j \in R_t : j \prec i\}\right| \tag{2.31}$$

3) Sort individuals in the merged population R_t according to a rank, afterward, select non-dominated individuals from R_t and put them into the archive A_{t+1}. If the size of A_{t+1} is more than archive size or Q then reduce it with truncation operator which is called in this thesis as CCMOA truncation. In the other hand, if the size of A_{t+1} is less than Q, judging in the ranks, the best Q - $|A_{t+1}|$ individual are considerably filled to A_{t+1}.

4) Perform binary tournament selection with replacement on archival individuals in order to fill the mating pool considered by ranks and summation of distances of a solution i, SDT_i, in archive and all solutions in the current mating pool. The binary selection of CCMOA will be described as follows. At beginning of selection, the summation of distances of a solution i in archive and all solutions in the current mating pool or SDT_i is set to 0. In the first selection, two individuals from archive are randomly picked; the individual with higher rank will be selected, otherwise if their ranks are equal, one individual is selected at random. After the selected individual is obtained, SDT_i of any solution i in the archive is updated by adding it with the distance between the solution i and this selected individual. In the next iteration, if ranks of two random individuals are not equal, one individual is selected as that of the first selection. If the ranks of these two individuals are equal, the

individual with more SDT_i is then selected. Subsequently, SDT_i of a solution i in the archive is updated again. The binary selection is then repeated until the mating pool is fulfilled

5) Apply CCMOA crossover and mutation operation, which will be described in the second following topic, within the mating pool. Then place the offspring in the population P_{t+1} and increase the generation counter by one ($t \leftarrow t + 1$).

6) Go back to step 2 until the termination condition is satisfied. Report the final archival individuals as the output solution set.

2.5.2. CCMOA truncation

From N non-dominated solutions, by CCMOA truncation, Q (archive size) solutions are obtained and placed into the archive A_{t+1}. The CCMOA truncation is as follows.

1) Find extreme solutions of the non-dominated solution. The number of extreme solution is equal to E, which is not exceed $2M$, (2 solutions represents for minimum and maximum value of each objective) as that of COGA-II. For instance, to find the extreme solution represents for minimum value of an objective k, if there is only one solution represents for the minimum value of the objective k, this solution is the extreme solution. Otherwise, if there are more than one solutions represent for the minimum value of the objective k, the solution is chosen at random to be the extreme solution.

2) All E extreme solutions are placed in the archive. Set the numbers of selected solutions $L = E$ and remaining solutions $R = N\text{-}E$. Then, calculate the Euclidean distance d_i^{RL} between the individual i in remaining solution set and its nearest neighbor in L. The d_i^{RL} for the remaining individual i is updated as COGA-II including a remaining individual j that have two-or-more individuals in the current archive with its same objective vector.

3) Remove the individual with highest value of d_i^{RL} from remaining solution set into the archive.

4) Increase the counter for the number of selected solutions ($L \leftarrow L + 1$) and decrease the counter for the number of remaining solutions ($R \leftarrow R - 1$). Update the d_i^{RL} for a remaining individual i.

5) Go back to step 3) until the number of selected individuals is equal to the required archive size.

2.5.3. CCMOA Crossover and Mutation

As previously stated, in the CCMOA a solution composes of several components or species. The reason for the success of the employment of co-operative co-evolution [43] as multi-objective optimizer [35] for multi-objective optimization problems which have weak linkage among decision variables or bits in a solution can be described as follows. A candidate solution, which may considerably be put into the archive, is a combined solution of which a part is obtained from an individual of any species while other parts are copied from a current archive solution. It can say that the candidate solution is not obtained by the crossover and mutation on the whole chromosome of parent individuals as that of normal MOEAs.

Due to the weak linkage of bits in a solution, the crossover and mutation on only a part of a solution without the change of other parts, which may already represent for good portions of the solution, can contribute more superior resulting solutions than that of the crossover and mutation on the whole chromosome as in normal MOEAs. Therefore the crossover and mutation of CCMOA employ only on one part of a solution on the corresponding partition of randomly selected species. For a problem for which a solution composes of S components or species represent by string of length L_1, L_2,..., L_S, respectively, two parents p_1 and p_2 will form their children c_1 and c_2. Figure 2.13 shows an example of CCMOA crossover and mutation of a problem with 4 species.

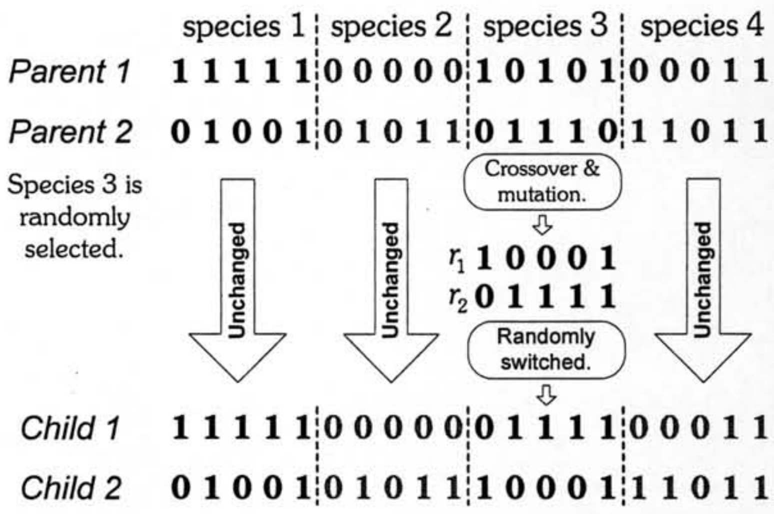
species 1
species 2
species 3
species 4
Parent 1 1 1 1 1 1 0 0 0 0 0 1 0 1 0 1 0 0 0 1 1
Parent 2 0 1 0 0 1 0 1 0 1 1 0 1 1 1 0 1 1 0 1 1
Species 3 is randomly selected.
Unchanged
Unchanged
Crossover & mutation.
r1 1 0 0 0 1
r2 0 1 1 1 1
Randomly switched.
Unchanged
Child 1 1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 0 0 0 1 1
Child 2 0 1 0 0 1 0 1 0 1 1 1 0 0 0 1 1 1 0 1 1

Figure 2.13 An example of CCMOA crossover and mutation of a problem with 4 species.

The crossover and mutation of CCMOA are described as follows.

1) Randomize species $i \in [1,S]$.

2) Copy corresponding chromosomes of length L_i of parents p_1 and p_2 onto temporary parents r_1 and r_2, respectively.

3) Apply crossover and mutation on r_1 and r_2 in usual way to obtain two resulting individuals t_1 and t_2, respectively.

4) Copy corresponding parts of other species of the parents p_1 and p_2 onto their children c_1 and c_2, respectively. Finally, t_1 and t_2 are randomly put onto the corresponding parts of species i of the children c_1 and c_2.

2.6. Improved Compressed-Objective Genetic Algorithm

Optimization problems usually arise when limited resources are available for existing demands. If multiple conflicting objectives are required in the problem formulation, the problem is multi-objective. Various techniques have been proposed for solving these multi-objective problems. Among these, the genetic algorithm has been established as one of the most widely used method for multi-objective optimization [27]-[33], [40]. Due to the parallel search nature of the algorithm, the approximation of multiple optimal solutions – the Pareto optimal solutions, comprising of non-dominated individuals – can be effectively executed. The performance of the algorithm always degrades as the search space or problem size gets bigger. An increase in the number of conflicting objectives has also significantly raised the difficulty level [39]. Thus, the non-dominated solutions may deviate from the true Pareto front; the coverage of the Pareto front by the solutions generated may be affected.

A number of strategies have been successfully integrated into genetic algorithms to solve these problems, including a direct modification of selection pressure [40], [42] and elitism [27], [31]. Although they have been proven to significantly improve the search performance of genetic algorithms, virtually all reported results deal with only few objectives. In reality, the possibility that a candidate solution is not dominated always increases with objective numbers, leading to an explosion in the total number of non-dominated solutions. This difficulty stems from the way that a non-dominated solution is defined. By Definition 2.1, domination definition, a candidate solution x is dominated by another solution y if and only if (a) all objectives from y are either better than or equal to the corresponding objectives from x and (b) at least one objective from y is better than the corresponding objective from x. Hence, if one single objective from y does not satisfy the conditions, y would not dominate x. In a problem with a large number of objectives, the chance that two solutions cannot dominate one another is inevitably high. A genetic algorithm must be able to pick out a well chosen solution set from a vast number of non-dominated solutions in order to successfully approximate the Pareto front.

In order to properly approximate such Pareto fronts, a number of non-dominated solutions must be excluded from the search target. One possible technique is to assign different preference levels to non-dominated solutions under consideration. It is hypothesized that a set containing highly preferred solutions would reflect a close approximation of the true Pareto front. The original compressed-objective genetic algorithm (COGA-I) [37] employed two conflicting criteria in the preference assignment. This can be viewed as a transformation from an *M*-objective problem to a two-objective problem during the survival competition between two non-dominated solutions. This thesis will present the improved version of compressed-objective genetic algorithm. The improved compressed objective genetic algorithm (COGA-II) is quite different from the original COGA-I in which three-or-more objectives is transformed to only one preference objective during the survival competition of two non-dominated solutions. The preference objectives of the COGA-I are winning score and vicinity index, in the other hand, the COGA-II has only one preference objective, winning score. Although the COGA-II employs winning score as the COGA-I, its winning score assignment is not the same as that of the original algorithm. The winning score assignment of both algorithms and the main procedure of the COGA-II will be described in the following topics.

2.6.1. Winning Score Assignment

When two non-dominated solutions are competing for survival, the solution with more winning score has a higher chance of being selected. The winning score assignments of COGA-I, and COGA-II will be described as the following topic.

1) Winning Score Assignment of COGA-I

In the original algorithm, COGA-I [37], the winning score is calculated from the numbers of superior and inferior objectives between a pair of two non-dominated solutions. Let sup_{ij} be the number of objectives in the solution i that is superior to the corresponding objectives in the solution j while inf_{ij} be the number of objectives in i that is inferior to j, for a population containing N non-dominated individuals, the winning score for the ith solution or WS_i is given by

$$WS_i = \sum_{j=1}^{N} w_{ij} \quad \text{where } w_{ij} = sup_{ij} - inf_{ij} \tag{2.32}$$

Obviously, $w_{ji} = -w_{ij}$ and $w_{ii} = 0$. With this assignment, non-dominated solutions with high winning scores should be close to the true Pareto front but they tend to cluster around the best values of individual objectives instead of spreading throughout the Pareto front. Therefore, the other preference objective, vicinity index, which is evaluated from the density of values in each objective, is introduced to prevent this problem. The non-dominated solutions are then assigned different ranks form their two preference objectives. However there may be strong conflict between these preference objectives for some non-dominated solutions, then the ranks of the solutions may be inappropriately assigned [37]. The improve compressed-objective genetic algorithm (COGA-II) eliminate this problem by using only winning score for which its winning score assignment, which is different from that of COGA-I, will be discussed in the next topic.

2) Winning Score Assignment of COGA-II

The winning score assignment of the original algorithm, COGA-I, which is described by the equation (2.32), employ the equal weight winning score, which is equal to 1, for any objective k. However, due to the relation between objectives of non-dominated solutions, for instance conflict, harmony, and independence [53], the weight winning scores of all objectives are not necessary equal to each other. For example, benchmark problems DTLZ5 and DTLZ6 [39] which have the same true Pareto front, for a set of true Pareto optimal solution of the problems, the relation of first M-1 objectives are harmony and the relation of these objective and M^{th} objective, the last objective, is conflict. Therefore, by the winning score assignment of the COGA-I, the true Pareto optimal solution with the best value M^{th} objective, which also has the worst first M-1 objectives, has the worst winning score which is much less than the winning score of the true Pareto optimal solution with the worst M^{th} objective, which also has the best first M-1 objectives. Since these solutions are the true Pareto optimal solutions, in a correct way they should have the same winning scores, then the winning scores of any non-dominated solutions in COGA-I may not be suitable to assign the level of the solutions.

To avoid this imperfection, the winning score assignment of COGA-II will employ the different weight winning score for any objective k which is evaluated from current non-dominated solution set. The winning score of a non-dominated solution is then calculated from the obtained weight winning scores. The weight winning score evaluation will be described as the follows. For N non-dominated solutions of an M-objective optimization problem, let n_{sup}, n_{inf}, and n_{eq} be the number of objectives in a solution i that is superior to the corresponding objectives in a solution j, the number of objectives in i that is inferior to j, and the number of objectives in i that is equal to j, respectively. The raw weight winning scores of objectives that i is superior to j and i is inferior to j are given by $\rho_{\sup}$ and $\rho_{\inf}$, then winning score of solution i to solution j or w_{ij} is given by $w_{ij} = n_{\sup}\rho_{\sup} - n_{\inf}\rho_{\inf}$. By consideration of the equality of preference level of solution i and solution j, it can be thought that $w_{ij} = w_{ji} = 0$ which implies that $n_{\sup}\rho_{\sup} = n_{\inf}\rho_{\inf}$. Let ρ_{ijk} be the raw weight winning score of solution pairs i and j of an objective k, by the consideration, the ρ_{ijk} can be given by the following equation.

$$\rho_{ijk} = \begin{cases} 0.5(n_{\sup} + n_{\inf})/n_{\inf}, & \text{for an objective } k \text{ that } i \text{ is superior to } j \\ 0.5(n_{\sup} + n_{\inf})/n_{\sup}, & \text{for an objective } k \text{ that } i \text{ is inferior to } j \\ 1, & \text{for an objective } k \text{ that } i \text{ is equal to } j \end{cases} \qquad (2.33)$$

The above equation implies that $\sum_{k=1}^{M} \rho_{ijk} = M$ for any solutions pair i and j. Thereafter, the summation of raw weight winning score of an objective k, f_k, of all possible solutions pair is given by

$$f_k = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \rho_{ijk} \tag{2.34}$$

Subsequently, the weight winning score of an objective k, $\overline{f}_k$, is defined by the following equation.

$$\overline{f}_k = \frac{f_i}{\sum f_k} \tag{2.35}$$

From the derivation, it can be see that $\overline{f}_k$ is the overall weighted winning score of a set of non-dominated solution of an objective k, and $\sum \overline{f}_k = 1$. For a population containing N non-dominated individuals, after $\overline{f}_k$ of any objective k is obtained, the winning score for the ith solution or WS_i is then given by

$$WS_i = \sum_{j=1}^{N} w_{ij}, \text{ where } w_{ij} = \sum_{k=1}^{M} \overline{f}_k q_{ijk} \tag{2.36}$$

The competitive score $q_{ijk} = 1$, if objective k of solution i is superior to that of solution j, $q_{ijk} = -1$, if objective k of solution i is inferior to solution j, and $q_{ijk} = 0$, if their objectives k are equal. In addition, for any solutions pair i and j, $w_{ji} = -w_{ij}$ and $w_{ii} = 0$, $-1 < w_{ij} < 1$, and $-N < WS_i < N$. By this winning score assignment, for a set of true Pareto optimal solution of the DTLZ5 or DTLZ6 problem the weight winning score of the M^{th} objective is M-1 times that of another objective, then the winning scores of any true Pareto optimal solutions are all equal. These examples show that the winning score assignment of COGA-II is more appropriate than that of the original COGA, COGA-I.

To study correction of the use of winning score to assign different level of non-dominated solutions, a number of non-dominated solutions are randomly generated, thereafter winning scores (WS_i) and distances (d_i) from true Pareto optimal front of the solutions are plotted by graph. The perfection of the use of winning score to that assignment can be indicated by the relation of WS_i and d_i, if this relation is illustrated by decreasing function, the winning score is a suitable indicator.

A minimization problem with M objectives of which an objective i or $f_i = x_i$ where $x_i \in [0, 1]$ is used as the example. This problem has only one true optimal point represented by the zero objectives. For the generation of a number of non-dominated solutions, the first random solution is generated and put into a non-dominated solution set A, after that another solution is randomly generated, if this solution neither dominates nor is dominated by any solution in set A, it is considerably put into set A. The generation is then repeated again until the set A is fulfilled. The following figure displays relation between winning score of solution i (WS_i), which is represent horizontal axis, and distance from the solution to the optimal solution, d_i, of a set of 200 non-dominated solution of the problem with 3-6, 8, 10, 15, 20 objectives respectively.

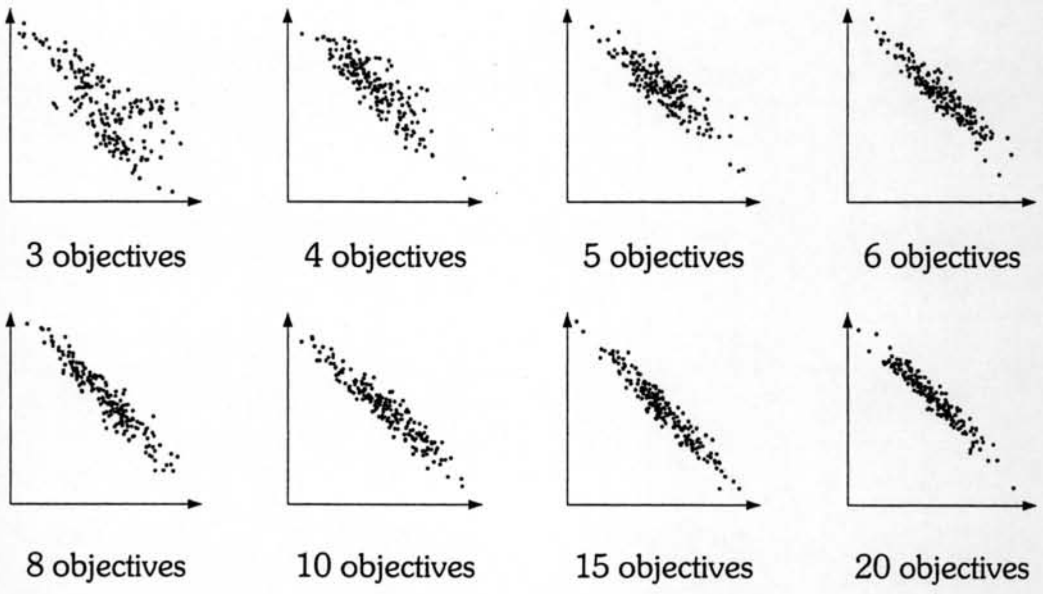

Figure 2.14 Relations between winning score (horizontal axis) and distance from the true solution (vertical axis).

Although, the figure shows that for any two solutions i and j, winning score of the solution i, WS_i, is more than that of the solution j, WS_j, it is not guarantee that the solution i is closer to the true solution than solution j, the overall relation between winning score and distance from the true solution can be described by the decreasing function. This shows that the winning score can used to approximate the quality of a solution in a non-dominated solution set. The figure also shows that if number of objectives is increased, the relation between the winning score and the distance is more similar to decreasing relation especially for the problems with 6, 8, 10, 15, and 20 objectives.

The employment of different weight winning score of any objective can prevent the bias of non-dominated solutions to cluster around the best values of individual objectives instead of spreading throughout the Pareto front. The prevention can be described as the following. If there are several solutions around the best objective for an objective *k*, the weight winning score $\overline{f}_k$ of the objective is less than that of the other objectives, and then an overall winning score of the solutions may be less than that of other solutions. Only a few solutions of this set are selected into mating pool, then, only some individuals of this set will be members of the archive for the next generation. As a result, this employment can prevent the bias of non-dominated solutions and obtain the solutions spread throughout the Pareto front.

3) Rank Assignment of Non-dominated Solutions

With the winning score, a maximization objective, a rank can be assigned to each non-dominated solution based upon the preference level as follows.

1) Evaluate a winning score of any solution of non-dominated solution set.

2) Find extreme solutions of the non-dominated solution. The number of extreme solution is equal to *E*, which is not exceed 2*M*, (2 solutions represents for minimum and maximum value of each objective). For instance, to find the extreme solution represents for minimum value of an objective *k*, if there is only one solution represents for the minimum value of the objective *k*, this solution is the extreme solution. Otherwise, if there are more than one

solutions represent for the minimum value of the objective k, the solution having the maximum winning score is chosen to be the extreme solution.

3) Sort E extreme non-dominated solutions in descending order of winning score, the first sorted solution is assigned with the rank of 1, second sorted solution is assigned with the rank of 2, and so on, therefore the lowest rank of extreme solutions is E. In the same way, N-E non-extreme non-dominated solutions are sorted in descending order of winning score, the first sorted solution is assigned with the next rank, $E+1$, and the second sorted solution is assigned with the rank of $E+2$ and so on. This rank assignment guarantees that a rank of an extreme solution is higher than that of any non-extreme solution.

2.6.2. COGA-II Main Procedure

The improved compressed-objective genetic algorithm (COGA-II) maintains the number of solutions in its archive to be constant which is different from the original algorithm, COGA-I. The reason of this improvement will be described as follows. In the COGA-I, the archive contains only non-dominated solutions which has the weak point in a problem for which a few non-dominated solutions in an early stage of a run. Then, there are only a few solutions in the archive, subsequently, after a selection (binary selection), many copies of a solution are put into the mating pool. This can cause premature convergence; to avoid this imperfection, the COGA-II will keep the number of solutions in archive constant. For the problem with a few non-dominated solutions in an early stage of a run, dominated solutions are filled into the archive until the number of solutions in the archive is equal to the defined size, after selection, only a few copies of a solution will be selected, therefore the mating pool will be then diversified. The procedure of the COGA-II, for which the archive size is equal to Q, is as follows.

1) Generate an initial population P_0 and an empty archive A_0. Initialize the generation counter ($t = 0$).

2) Merge the population P_t and the archival A_t together to form a merged population R_t. After V non-dominated individuals and W dominated individuals of the merged population R_t are obtained, all V non-dominated individuals are put into the archive A_{t+1}. If $V > Q$, truncate the individual set using the operator described in the following topic. If $V = Q$, go to the next step, else if $V_t < Q$, the Q-V dominated individuals having the least number of their dominators are therefore fulfilled the current archive A_{t+1}.

3) Evaluate the rank of each individual of non-dominated solutions in A_{t+1} by the rank assignment of non-dominated solutions as previously described, while the rank of each dominated individual in A_{t+1} is equal to V_t plus number of its dominators in R_t. The addend V_t ensures a rank of a non-dominated individual is higher than that of a dominated individual.

4) Perform binary tournament selection with replacement on archival individuals in order to fill the mating pool. The binary selection of COGA-II will be described as follows. For each selection, two individuals from archive are randomly picked; the individual with higher rank will be selected. The binary selection is then repeated until the mating pool is fulfilled.

5) Apply crossover and mutation operation within the mating pool. Then place the offspring in the population P_{t+1} and increase the generation counter by one ($t \leftarrow t + 1$).

6) Go back to step 2 until the termination condition is satisfied. Report the non-dominated solution set of the final archive as the output solution set.

A truncation operator for maintaining individuals in the archive is introduced next.

2.6.3. Truncation Operator for the Archive

For a set of *N* non-dominated solutions, the truncation for the archive with size *Q* is based upon the preference level, which is indicated by winning score, and the spread of solutions in the objective space. It can be described as follows.

1) Calculate winning scores of all *N* non-dominated individuals.

2) Find extreme solutions by the determination of an extreme solution in the rank assignment of non-dominated solutions, which is earlier described in the previous topic. Thereafter, all *E* extreme solutions are placed in the archive. Set the numbers of selected solutions $L = E$ and remaining solutions $R = N\text{-}E$ and then calculate the Euclidean distance d_i^{RL} between an individual *i* in remaining solution set and the nearest neighbor in the archive.

3) Select $(Q - L)$ solutions with highest values of d_i^{RL} from the remaining solution set. From these selected solutions, the candidate with the most winning score is moved to the archive. If there are more than one solutions having the most winning score, the suitability is decided by the high value of d_i^{RL}.

4) Increase the counter for the number of selected solutions $(L \leftarrow L + 1)$ and decrease the counter for the number of remaining solutions $(R \leftarrow R - 1)$. The d_i^{RL} for the remaining individual *i* is updated in the usual way. In the other hand, for an individual *j* in the remaining individual set that have two-or-more individuals in the current archive with the same objective vector as the individual *j*, the d_i^{RL} of the individual *j* is set by the following equation.

$$d_j^{RL} = -(C_j - 1)\sqrt{M} \tag{2.37}$$

where C_j is the number of individuals in the current archive that have the same objective vector as the individual *j*. This setting guarantee that for any two individuals *i* and *j* in which C_i is less than C_j, the distance d_i^{RL} is certainly more than d_i^{RL}, then it can prevent the occurrence of several individuals having a same objective vector in the archive.

5) Go back to step 3 until the number of selected individuals is equal to the required archive size.

Note that after the archive is obtained, solutions in archive will be assigned ranks; their winning scores are also calculated in the rank assignment, therefore the winning score of a solution i, WS_i, evaluated in the truncation is not equal to WS_i, evaluated in the rank assignment because they are calculated from the different sets of solution.

2.7. Co-operative Co-evolutionary Improved Compressed-Objective Algorithm

A co-operative co-evolutionary improved compressed-objective genetic algorithm (CCCOGA-II) is the resulting algorithm of the integration of an improved compressed-objective genetic algorithm (COGA-II) and a co-operative co-evolutionary multi-objective algorithm (CCMOA). The main procedure of CCCOGA-II is the same as that of COGA-II, only crossover and mutation of CCCOGA-II is different from COGA-II; it is the same as that of CCMOA.