Finding Acceptable Pareto-Optimal Solutions using Multiobjective Genetic Algorithms

P. J. Bentley[1] **and J. P. Wakefield**[2]

[1]*Department of Computer Science, University College London, Gower Street, London WC1E 6BT, UK.*
Tel. 0171 391 1329 P.Bentley@cs.ucl.ac.uk (corresponding author)
[2]*Pro:Technica Consultants Limited, Oak House, Ransom Wood Park, Southwell Road West, Mansfield, Notts. NG21 0HJ. England*

Keywords: multiobjective optimization, Pareto-optimal distributions, acceptable solutions, genetic algorithm

Abstract

This paper investigates the problem of using a genetic algorithm to converge on a small, user-defined subset of *acceptable* solutions to multiobjective problems, in the Pareto-optimal (P-O) range. The paper initially explores exactly why separate objectives can cause problems in a genetic algorithm (GA). A technique to guide the GA to converge on the subset of acceptable solutions is then introduced.

The paper then describes the application of six multiobjective techniques (three established methods and three new, or less commonly used methods) to four test functions. The previously unknown distribution of solutions produced in the P-O range(s) by each method is described. The distribution of solutions and the ability of each method to guide the GA to converge on a small, user-defined subset of P-O solutions is then assessed, with the conclusion that two of the new multiobjective ranking methods are most useful.

1. Introduction

The genetic algorithm (GA) has been growing in popularity over the last few years as more and more researchers discover the benefits of its adaptive search. Many papers now exist, describing a multitude of different types of genetic algorithm, theoretical and practical analyses of GAs and huge numbers of applications for GAs [7,8]. A substantial proportion of these applications involve the evolution of solutions to problems with more than one criterion. More specifically, such problems consist of several separate objectives, with the required solution being one where some or all of these objectives are satisfied to a greater or lesser degree. Perhaps surprisingly then, despite the large numbers of these multiobjective optimization applications being tackled using GAs, only a small proportion of the literature explores exactly how they should be treated with GAs.

With single objective problems, the genetic algorithm stores a single fitness value for every solution in the current population of solutions. This value denotes how well its corresponding solution satisfies the objective of the problem. By allocating the fitter members of the population a higher chance of producing more offspring than the less fit members, the GA can create the next generation of (hopefully better) solutions. However, with multiobjective problems, every solution has a number of fitness values, one for each objective. This presents a problem in judging the overall fitness of the solutions. For example, one solution could have excellent fitness values for some objectives and poor values for other objectives, whilst another solution could have average fitness values for all of the objectives. The question arises: which of the two solutions is the fittest? This is a major problem, for if there is no clear way to compare the quality of different solutions, then there can be no clear way for the GA to allocate more offspring to the fitter solutions.

The approach most users of GAs favour to the problem of ranking such populations, is to weight and sum the separate fitness values in order to produce just a single fitness value for every solution, thus allowing the GA to determine which solutions are fittest as usual. However, as noted by Goldberg: "...there are times when several criteria are present simultaneously and it is not possible (or wise) to combine these into a single number." [7]. For example, the separate objectives may be difficult or impossible to manually weight because of unknowns in the problem. Additionally, weighting and summing could have a detrimental effect upon the evolution of acceptable solutions by the GA (just a single incorrect weight can cause convergence to an unacceptable solution).

Moreover, some argue that to combine separate fitnesses in this way is akin to comparing completely different criteria; the question of whether a good apple is better than a good orange is meaningless.

The concept of Pareto-optimality helps to overcome this problem of comparing solutions with multiple fitness values. A solution is Pareto-optimal (i.e., Pareto-minimal, in the Pareto-optimal range, or on the Pareto front) if it is *not dominated* by any other solutions. As stated by Goldberg [7]:

Definition 1. A vector $\mathbf{x}$ is partially less than $\mathbf{y}$, or $\mathbf{x} <_p \mathbf{y}$ when:
$(\mathbf{x} <_p \mathbf{y}) \Leftrightarrow (\forall_i)(\mathbf{x}_i <= \mathbf{y}_i) \wedge (\exists_i)(\mathbf{x}_i < \mathbf{y}_i)$

$\mathbf{x}$ *dominates* $\mathbf{y}$ iff $\mathbf{x} <_p \mathbf{y}$.

However, it is quite common for a large number of solutions to a problem to be Pareto-optimal (and thus be given equal fitness scores). This may be beneficial should multiple solutions be required, but it can cause problems if a smaller number of solutions (or even just one) is desired.

For example, consider the multiobjective function (to be minimised):

$$f_1 = (x + 50)^2$$
$$f_2 = (x - 50)^2 \qquad \text{where } -64 \le x \le 64.$$

For this twin-objective function, the Pareto-minimal solutions range from -50 to 50 — so almost every allowable value of x is Pareto-optimal, see fig. 1. Although this is an extreme case, it does illustrate a fundamental flaw with the concept of Pareto-optimality: the Pareto front can be so large that it becomes infeasible to use the non-dominance of solutions as the sole fitness measure for solutions in a GA.

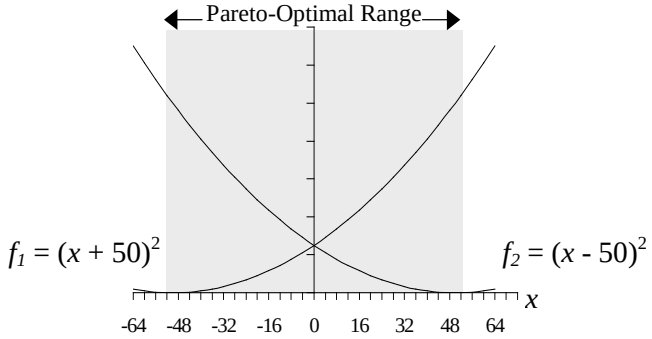


Figure 1. The Pareto-optimal range of solutions for some multiobjective functions can include almost all allowable solutions to the problem.

Hence, for many problems, the set of solutions deemed acceptable by a user will be a small sub-set of the set of Pareto-optimal solutions to the problems [4]. Manually choosing an acceptable solution can be a laborious task, which would be avoided if the GA could be directed by a ranking method to converge only on acceptable solutions. For this work, an *acceptable solution* (or champion solution) is defined:

Definition 2. A solution is an *acceptable solution* if it is Pareto-optimal and it is considered to be acceptable *by a human*.

Consequently, this paper will investigate the problem of using a genetic algorithm to converge on a small, user-defined subset of acceptable solutions to multiobjective problems, in the Pareto-optimal (P-O) range.

The paper will initially focus on the difficulties posed by multiobjective problems to genetic algorithms. A technique to guide the GA to converge on the smaller subset of acceptable solutions will then be introduced. In the light of this, six different ranking methods will be described: three commonly used methods ('sum of weighted objectives', 'non-dominated sorting', and 'weighted maximum ranking' - based on Schaffer's VEGA [11]), and three new, or less commonly used methods ('weighted average ranking', 'sum of weighted ratios', and 'sum of weighted global ratios').

This paper will then describe the application of these six multiobjective techniques to four established test functions, and will examine the previously unexplored distribution of solutions produced in the P-O range(s) by each method. The distribution of P-O solutions and the ability of each method to guide the GA to converge on a small, user-defined subset P-O solutions will then be assessed.

2. Background

Existing literature seems to approach this ranking problem using methods that can be classified in one of three ways: the aggregating approaches, the non-Pareto approaches and the Pareto approaches.

Many examples of aggregation approaches exist, from simple 'weighting and summing' [7,15] to the 'multiple attribute utility analysis' (MAUA) of Horn and Nafpliotis [9]. Of the non-Pareto approaches, perhaps the most well-known is Schaffer's VEGA [11,12], who (as identified by Fonseca [3]) does not *directly* make use of the actual definition of Pareto-optimality. Many other non-Pareto methods have been proposed (e.g. by Linkens [5], Ryan [10] and Sun [14]). Finally the Pareto-based methods, proposed first by Goldberg [7] have been explored by researchers such as Horn [9] and Srinivas [13].

In addition, many researchers are now introducing 'species formation' and 'niche induction' in an attempt to allow the uniform sampling of the Pareto set (e.g. Goldberg [7] and Horn [9]). For a comprehensive review, see the paper by Fonseca and Fleming [3].

3. Range-Independence

Since the searching mechanism of the genetic algorithm is based upon that of natural evolution, nature is the first logical place to look for methods of evolving fit multiobjective individuals.

In nature, every living creature must satisfy a large number of objectives sufficiently in order to be successful (e.g. avoid predators, find food, survive illnesses, reproduce). However, nature cannot (and has no need to) determine precisely which creature will be more successful (or fitter) than another. Whilst nature does sometimes attempt to improve the chances of successful creatures passing on their genes by allowing the physically stronger members of a group to breed more than weaker ones (e.g. male lions, deer, walruses), criteria such as physical strength can only be indirect approximations to the overall fitness (ability to produce good offspring) of the creature. Indeed, the criteria used for mating selection sometimes seems to bear little relation to the fitness of the creature (e.g. the tail of the peacock). This does not matter greatly, for in nature, the definition of a 'successful' creature is one that has managed to become the ancestor of a long lineage of similar offspring.

With genetic algorithms, however, a solution is not considered 'fit' if it has produced good offspring - quite the reverse: a solution is allocated a greater chance of having offspring if it is identified as being fit. In these algorithms, fitness now becomes a measure of how well the solution satisfies mathematical objectives and bears little relation to the real measure used in nature. Therefore, because a mathematical function is being used to judge fitness, this problem of ranking multiobjective solutions is purely artificial.

Consequently, the problem has more to do with mathematics than nature. Throughout the evolution by the GA, every separate objective (fitness) function in a multiobjective problem will return values within a particular range. Although this range may be infinite in theory, in practice the range of values will be finite. This 'effective range' of every objective function is determined not only by the function itself, but also by the domain of input values that are produced by the GA during evolution. These values are the parameters to be evolved by the GA and their exact values are normally determined initially by random, and subsequently by evolution. The values are usually limited still further by the coding used, for example 16 bit sign-magnitude binary notation per gene only permits values from -32768 to 32768. Hence, the *effective range* of a function can be defined:

Definition 3.	The *effective range* of $f(x)$ is the range from $\min(f(x))$ to $\max(f(x))$ for all values of x that are actually generated by the GA, and for no other values of x.

Although occasionally the effective range of all of the objective functions will be the same, in most more complex multiobjective tasks, every separate objective function will have a different effective range (i.e., the function ranges are noncommensurable [12]). This means that a bad value for one could be a reasonable or even good value for another, see fig. 2. If the results from these two objective functions were simply added to produce a single fitness value for the GA, the function with the largest range would dominate evolution (a poor input value for the objective with the larger range makes the overall value much worse than a poor value for the objective with the smaller range).

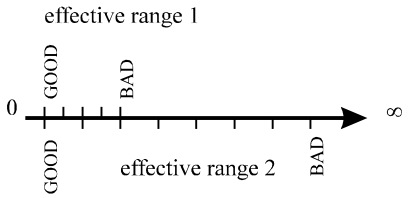


Figure 2. Different effective ranges for different objective functions (to be minimized)

For example, consider the two objective functions:

$f_{11} = x^2$

$f_{12} = (x - 2)^2 / 1000$

(both to be minimised).

Given a non-optimal input value, the output value from f_{11} will normally be three orders of magnitude worse than that from f_{12} (i.e., the second function will be approximately one thousand times closer to the minimum of zero). As can be seen in the simplest of tests, if the outputs from both were simply summed, the first function would completely dominate the second, resulting in the effective evolution of a good solution only to the first function.

Thus, the only way to ensure that all objectives in a multiobjective problem are treated equally by the GA is to ensure that all the effective ranges of the objective functions are the same (i.e., to make all the objective functions commensurable), or alternatively, to ensure that no objective is directly compared to another. In other words, either the effective ranges must be converted to make them equal, and a range-dependent ranking method used, or a range-independent ranking method must be used. Typically, range-dependent methods (e.g., 'sum of weighted objectives', 'distance functions', and 'min-max formulation') require knowledge of the problem being searched to allow the searching algorithm to find useful solutions [13]. Range-independent methods require no such knowledge, for being independent of the effective range of each objective function makes them independent of the nature of the objectives and overall problem itself. Hence, a ranking method should not just be independent of individual applications (i.e., problem independent), as stated by Srinivas [13], it should be independent of the effective ranges of the objectives in individual applications (i.e., range-independent). Multiobjective ranking methods that are *range-dependent* or *range-independent* can be defined:

Definition 4. Given the objective functions of a problem: $f_{1..n}(x)$
and a set of solution vectors to the problem: $\{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_m\}$

A multiobjective ranking method is *range-dependent* if the fitness ranking of $\{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_m\}$ defined by the method changes when the effective ranges of $f_{1..n}(x)$ change.

A multiobjective ranking method is *range-independent* if the fitness ranking of $\{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_m\}$ defined by the method *does not* change when the effective ranges of $f_{1..n}(x)$ change.

For example, the standard 'sum of weighted objectives' method favoured by so many, uses the weights to make the effective domains of each objective equal, then provides a single fitness value by summing the resulting values. This is a range-dependent method, for it relies completely on the weights being set precisely for every problem. Should any of the objectives be changed, or the allowable domain of input values be changed (perhaps by a change in coding, or seeding the initial population with anything other than random values), then these weights may have to be changed.

Alternatively, the non-dominated sorting method, and variants of it, is a range-independent method. It requires no weighting of the objective values, for the fitness values from each objective function are never directly compared with each other. Only values from the same objective are ever compared in the process of determining the non-dominance of solutions (Goldberg 1989). For complex multiobjective problems, this range-independence is extremely advantageous: good results do not depend on the ability of the user to fine-tune weights correctly. However, a disadvantage of non-dominated sorting is that all Pareto-optimal solutions are considered equally good, regardless of what the user actually regards as being acceptable.

4. Importance

In addition to being range-independent, there is another significant, and usually overlooked property that a good ranking method should have: the ability to increase the 'importance' of some objectives with respect to others in the ranking of solutions, to allow search to be directed to converge on *acceptable* solutions. *Importance* can be defined:

Definition 5. *Importance* is a simple way to give a ranking method additional problem-specific information, in order to direct a GA to converge on acceptable solutions within a smaller subset of the Pareto-optimal range, by favouring those solutions closer to the optima of functions with increased importance, in proportion to this increased importance.

It has been known for some time that the quality of solutions to complex search problems can be improved by increasing the importance of a particular part or objective of the problem [2,6]. This is often achieved either by introducing objectives to the search algorithm one at a time (or in distinct 'stages') with the most important first, or by simply weighting the most important objectives more heavily. Indeed, experience shows that many users of GAs and the 'sum of weighted objectives' ranking method are inadvertently increasing the importance of certain objectives without being aware of it, as they fine-tune their weights to improve evolution. In other words, the dual nature of these weights (i.e., the fact that each weight can not only equalise the effective ranges of objectives, but also define increased importance for objectives), is often overlooked.

Intentionally determining which objectives are more important in a problem can be a matter of debate, but to improve evolution time, it seems that often the best results are gained by making the most difficult-to-satisfy objectives the most important. However, some problems demand that certain objectives have differing levels of importance just to allow evolution of an acceptable solution. (For example, the optimization of an electronic device has the design criteria: cost, speed, size and power consumption. For some devices, a low cost is overwhelmingly important, for others, a high speed is of greatest importance.)

Significantly, 'importance' can be used regardless of what the individual objective functions represent. Hence, objective functions that represent wildly different things can be judged against each other. Often, when two functions represent different things, despite any similarity between their effective ranges, they are, perhaps improperly, called noncommensurable. To clarify this term, for this work, *commensurable* functions are defined:

Definition 6. Two or more functions are *commensurable* if the difference between the effective ranges of the functions is insignificant (i.e., the differences between the minimum and maximum of each must be negligible), *regardless of what these functions represent*.

Hence, given a problem with two commensurable objective functions, whatever each one represents (be it cars or carrots), solution vectors to the problem can have their fitnesses precisely set, using the relative importance values of the objectives.

To illustrate this, consider the problem of packing a bag before going mountaineering. In this simplified example, the person has to choose between the amount of climbing equipment and the amount of food to be packed in the bag. How much of each should be packed? Many researchers would state that the two are non-commensurable and cannot be directly compared by a computer, and so would present a human user with a number of alternative solutions to choose from. Clearly, in reality, the ideal solution depends on the length of time of the trip, and the difficulty of the climb. If the trip is to take two days, and will involve only a hike in some hills, then more food is required than climbing equipment. However, if the climb will involve an hour scaling a vertical cliff, then more climbing equipment is required than food. In other words, a human picks a solution based on the relative *importance* of the two objectives. Moreover, there is no good reason not to specify these relative importance values for the computer, and let the computer pick the *same solution* (without the need for a human to consider potentially hundreds of different Pareto-optimal solutions).

Consequently, it is clear that importance is an essential tool to help the evolution of acceptable solutions. What is perhaps less clear, is how the concept of importance should be implemented within multiobjective ranking methods.

One way to allow the definition of importance within aggregation-based ranking methods is to take advantage of the fact that these methods usually guide the GA to converge upon a single 'best compromise' solution. For the purposes of this paper, the *best compromise* solution is defined:

Definition 7. A *best compromise* solution is the solution with the sum of (weighted) objective fitnesses minimized.

By weighting appropriate objectives with importance values, this best compromise solution can be made the same as (or at least moved into the vicinity of) the required solution, allowing the GA to converge directly to an acceptable solution. Thus, producing a single best compromise solution is not always a disadvantage.

Nevertheless, the more favoured ranking methods do not employ aggregation (and typically are range-independent). They are usually used with some form of niching and speciation method to allow the GA to generate not one, but a range of non-dominated P-O solutions. (Niching can also help the quality of solutions by preventing excessive competition between distant solutions [7].) The user is then required to select the preferred solution from this range of different solutions.

However, particularly for problems with many objectives, only a small proportion of P-O solutions may be acceptable solutions. This means that even when hundreds of different solutions are generated by the GA, there can be no guarantee that an acceptable solution will be among them. Moreover, for such large problems, it is not always feasible to allow the user to pick the preferred solution from a truly representative range of P-O solutions: the number to be considered may be too large. Thus, the ranking method needs further information, to guide the algorithm to converge more closely to acceptable solutions *within* the range of P-O solutions. This information is 'importance' - by specifying which objectives must be satisfied more than others, the GA can converge more closely to acceptable solutions, not just P-O solutions.

Unfortunately, there is no easy way to increase the importance of one objective in relation to another, without the two objectives being directly compared to each other. In other words, whilst it is simple to specify increased importance with a range-dependent aggregation method such as 'sum of weighted objectives' (just increase the weights), with a range-independent method such as 'non-dominated sorting', specifying importance is more complex. (Fonseca forces a kind of importance with his 'preference articulation' method [4], but this requires detailed knowledge of the ranges of the functions themselves, and is not a continuous guide to evolution.) Thus, alternative methods of ranking multiobjective solutions are required, that are ideally range-independent and allow the easy specification of importance, to enable the GA to converge on the subset of acceptable solutions.

5. Multiobjective Ranking Methods

There follows descriptions of six different ranking methods. The first three are the most commonly used methods: the range-dependent 'weighted sum' (aggregation) method, the range-independent Pareto non-dominated sorting, and a range-independent method based on Schaffer's VEGA [11,12]. The last three are new range-independent methods, developed in an attempt to allow importance to be specified with such methods. The techniques used within these methods are not new, but they have as yet been rarely used to rank multiobjective populations within a genetic algorithm. (Algorithms are included in the appendix.)

Method 1: Sum of Weighted Objectives (SWO)
This is perhaps the most commonly used method because of its simplicity. All separate objectives are weighted to make the effective ranges equivalent (and to specify importance) and then summed to form a single overall fitness value for every solution. These values are then used by the GA to allocate the fittest solutions a greater chance of having more offspring. (Because of the similarity in nature and performance between this method and many of the other 'classical' methods [13], only this classical method will be explored.)

Method 2: Non-Dominated Sorting (NDS)
Described by Goldberg [7], this range-independent method and variants of it are commonly used. The fitnesses of the separate objectives are treated independently and never combined, with only the value for the same objective in different solutions being directly compared. Solutions are ranked into 'non-dominated' order, with the fittest being the solutions dominated the least by others (i.e., having the fewest solutions partially less than themselves). These fittest can then be allocated a greater probability of having more offspring by the GA.

Method 3: Weighted Maximum Ranking (WMR)
This ranking method is based on Schaffer's VEGA [11,12]. WMR forms lists of fitness values of each solution for each objective. The fittest *n* solutions from each list are then extracted, and random pairs are selected for reproduction. Importance levels can be set by weighting appropriate fitness values for solutions. Note that the additional heuristic used by Schaffer to encourage 'middling' values [11] was not implemented in WMR.

Method 4: Weighted Average Ranking (WAR)

This is the first of the alternative ranking methods proposed. The separate fitnesses of every solution are extracted into a list of fitness values for each objective. These lists are then individually sorted into order of fitness, resulting in a set of different ranking positions for every solution for each objective. The average rank of each solution is then identified, with this value allowing the solutions to be sorted into order of best average rank. Thus, the higher an average rank a solution has, the greater its chance of producing more offspring. Since all objective fitnesses are treated separately, this method is range-independent. This technique allows the specification of importance by the weighting of average ranking values for each solution.

Method 5: Sum of Weighted Ratios (SWR)

This is the second of the ranking methods proposed for GAs and is basically an extension to SWO (method 1). The fitness values for every objective are converted into ratios, using the best and worst solution in the current population for that objective every generation. More specifically:

$$fitness\_ratio_i = \frac{(fitness\_value_i - \min(fitness\_value))}{(\max(fitness\_value) - \min(fitness\_value))}$$

This removes the range-dependence of the solutions, and they can be weighted (for the setting of importance) and summed to provide a single fitness value for each solution as with the first method.

Method 6: Sum of Weighted Global Ratios (SWGR)

This method is the third of the proposed ranking methods for GAs, and is a variation of SWR (method 5). Instead of the separate fitnesses for each objective in every solution being converted to a ratio using the *current* population best and worst values, the *globally* best and worst values are used. Again the importance of individual objectives can be set by weighting the appropriate values.

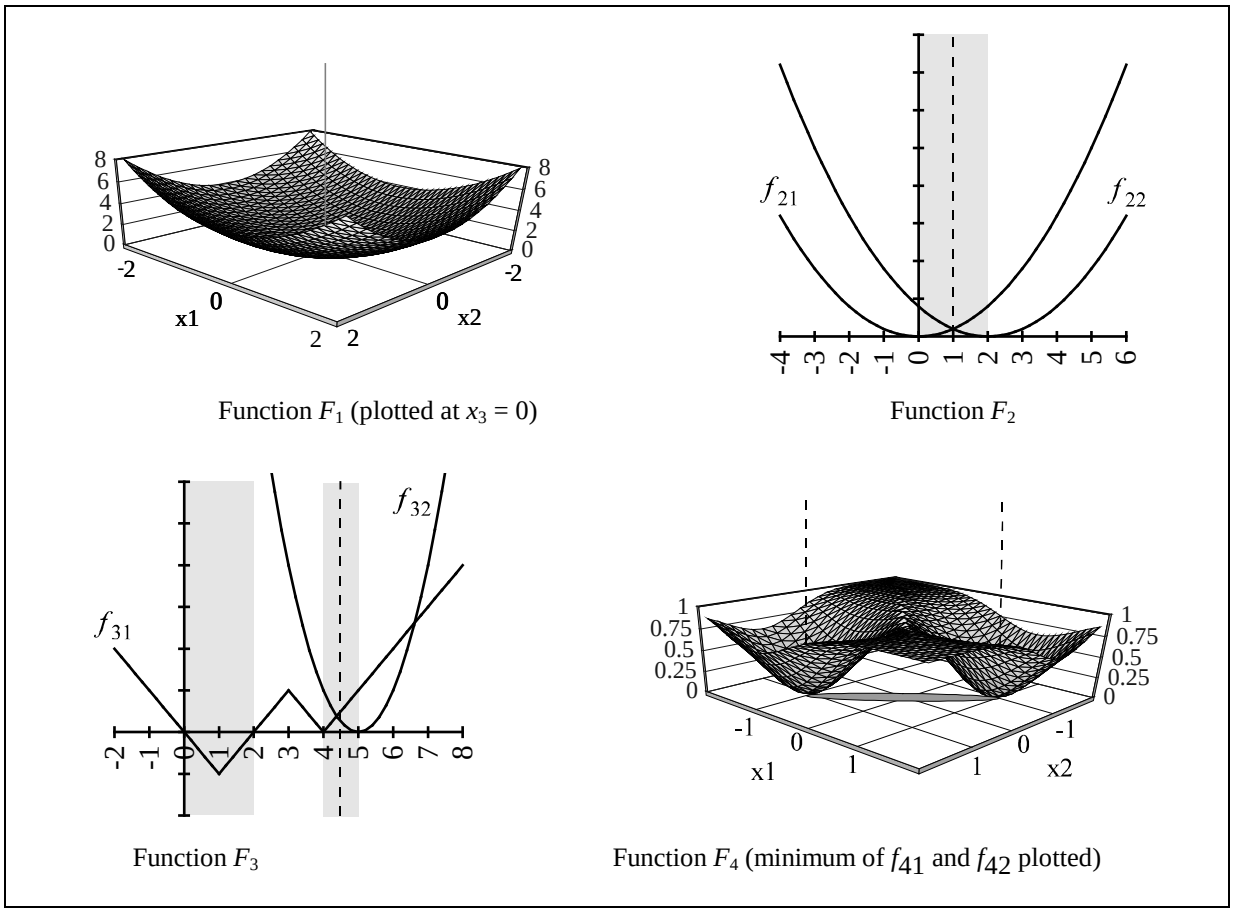
6. Application of the Ranking Methods

6.1. Test Functions

To explore and compare the distributions of solutions generated by the six ranking methods, they were applied in turn to four different test functions: F_1 to F_4. The first three are identical to those used by Schaffer [11,12], whilst F_4 is identical to Fonseca's f_1 [4]. Each function was chosen to represent a different class of function (i.e., each has different numbers of P-O ranges and/or best compromise solutions). All functions are to be minimized, see Table 1 and Fig. 3.

Function F_1 f_1 = $x_1^2 + x_2^2 + x_3^2$	A single-objective, three parameter function with an optimal at (0,0,0).
Function F_2 f_{21} = x^2 f_{22} = $(x - 2)^2$	A simple twin-objective single parameter function with a P-O range between 0 and 2, and a best compromise solution of 1.
Function F_3 f_{31} = -x where x <= 1 -2 + x where 1 < x <= 3 4 - x where 3< x <= 4 -4 + x where 4 < x f_{32} = $(x - 5)^2$	A twin-objective single parameter function with two disjoint P-O ranges: 0 to 2 and 4 to 5. This has a single best compromise solution of 4.5
Function F_4 f_{41} = $1 - \exp(-(x_1 - 1)^2 - (x_2 + 1)^2)$ f_{42} = $1 - \exp(-(x_1 + 1)^2 - (x_2 - 1)^2)$	A twin-objective function this time with two parameters. This has a single P-O range along the line (-1,1) to (1,-1), but two equal best compromise solutions at the optima of each function: (-1,1) and (1,-1).

Table 1 The four test functions used to compare the ranking methods.



Function F_1 (plotted at $x_3 = 0$)

Function F_2

Function F_3

Function F_4 (minimum of f_{41} and f_{42} plotted)

Figure 3. Graphs of the four test functions with Pareto-optimal ranges shown by grey shaded regions and best compromise solutions marked with dotted lines (P-O ranges shown by grey shaded regions and best compromise solutions marked with dotted lines).

All six methods were used with a basic genetic algorithm using binary coding, a population size of 50, and running for 100 generations. Probability of crossover was 1.0, probability of mutation was 0.01. Although this GA used elitist selection techniques, with all of the ranking methods described in this paper it is possible to use alternatives.

The distributions produced by methods 1-6 for each function were calculated by running the GA between 1,000 and 10,000 times (1000 runs for F_1, 2000 runs for F_2 and F_3, and 10000 runs for F_4). It was assumed that the distribution of solutions produced by a series of runs of this algorithm would not differ significantly from the distribution of solutions obtained by an algorithm with niching or other speciation techniques.

6.2. Evolved Results: F_1

The first experiment performed with each method was simply to allow the GA to minimize F_1. This function was used to validate that each method would rank solutions to single-objective problems correctly (as was done for VEGA by Schaffer [12]. As expected, every method allowed the GA to converge on, or very near to, the optimal solution of (0,0,0), every time. (The distributions of solutions for this function are all at a single point and hence are not shown.)

6.3. Evolved Results: F_2

The next experiment involved minimising F_2. To give some idea of the quality and distribution of solutions, 2,000 test runs were performed for each method. All methods allowed the GA to produce P-O solutions every time, however, as fig. 4 shows, the distribution of these solutions on the Pareto front for this function are very different for each method. SWO and SWGR both produced solutions very close to or exactly the best compromise value of 1.0. SWR also favoured this value, but with a larger 'spread', with the numbers of solutions produced falling almost logarithmically the further from the best compromise value they were. NDS showed a fairly even distribution throughout the P-O range, and WMR favoured solutions at either function optima, with nothing in between. WAR gave the most unexpected and fascinating distribution, with solutions close to each optima and close to the best compromise value being favoured, all other P-O values being less commonly produced, see fig. 4.

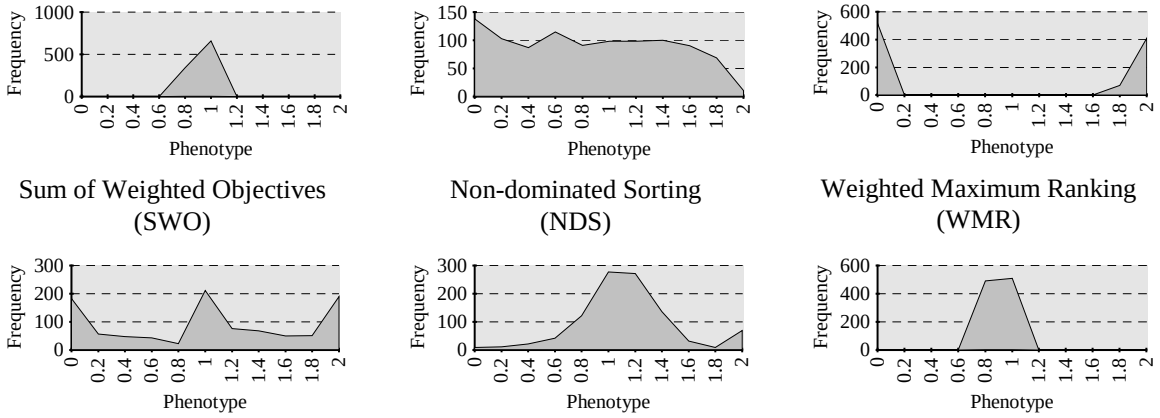


Figure 4. Distributions of solutions within the Pareto-optimal range for function F_2.

Additionally for F_2, the average solution of each method was calculated to give an indication of how balanced these distributions were. In other words, no matter what value(s) of P-O solution were favoured, the mean value for F_2 should be the centre value of 1.0. Table 2 (F_2 *test 1*) shows that all methods produced mean solutions close to 1.0.

	Best Compromise	SWO	NDS	WMR	WAR	SWR	SWGR
F_2 *test 1*	1.0	1.00922	0.93999	0.97595	1.10226	1.21556	0.98763
F_2 *test 2*	1.0	2.01459	0.85992	0.99532	1.17007	1.22672	0.98825
F_2 *test 3*	1.333	1.37837	N/A	1.45757	2.01466	1.66141	1.310

Table 2. Average solutions for each ranking method in F_2 tests 1-3.

Two further tests were performed using F_2. For the second test, f_{21} was temporarily changed to:
$f_{21} = x^2 / 1000$
to investigate the range-independence (or lack of it) for each method. As Table 2 (F_2 *test 2*) shows, after 2000 test runs for each method, SWO (method 1) clearly demonstrates its range-dependence by converging, on average, to the optimal of f_{22} instead of near to 1.0. All other methods show their range-independence by continuing to give mean solution values close to 1.0.

Finally, for the third test with F_2, the importance of f_{22} was doubled for every method capable of supporting importance (the two objectives being otherwise unchanged from the first test). By increasing the importance, the best compromise solution (i.e., the minimum of weighted and summed objectives) is changed from 1.0 to 1.333. Only three methods: SWO, SWR and SWGR, all successfully produced values close to this new desired value (see Table 2, F_2 *test 3*). NDS does not support importance, and WMR just doubled the frequency of optimal solutions to f_{22} (giving a deceptive mean solution), without actually producing any values between the two function optima. Finally, and quite unexpectedly, WAR simply converged every time to the optimal of f_{22}.

Upon investigation, it emerged that WAR does not permit the specification of gradual importance values. It was expected that increasing the weighting of the ranking value for more important objectives would introduce some level of additional importance for these objectives. Interestingly though, in practice it does not appear to be possible to gradually increase 'importance' values for this method: either all objectives are treated equally, or the objective with the increased weight dominates all other objectives completely. Somewhat counter-intuitively, it seems that no matter how large or small an increase is made to a weight, that objective will dominate all others.

6.4. Evolved Results: F_3

Experiments were then performed using F_3 with each method in turn. The function F_3 is significant since it has two disjoint P-O ranges. Nevertheless, the distributions of solutions for this function were surprisingly consistent with those for F_2, see fig. 5. As before, SWO and SWGR almost always converged to solutions near to the best compromise value of 4.5 (for F_3). Again, SWR favoured the best compromise solution with a slightly larger 'spread', but this time some solutions close to the optimal of f_{31} were also produced. NDS gave a fairly even distribution of solutions within the two P-O ranges, and WMR again only generated solutions at the optima of the two objectives, with none in between. Finally, WAR showed its highly unusual distribution once more, by favouring solutions close to the optima of both objectives (including both minima of the multimodal objective f_{31}), and the best compromise solution to a lesser degree.

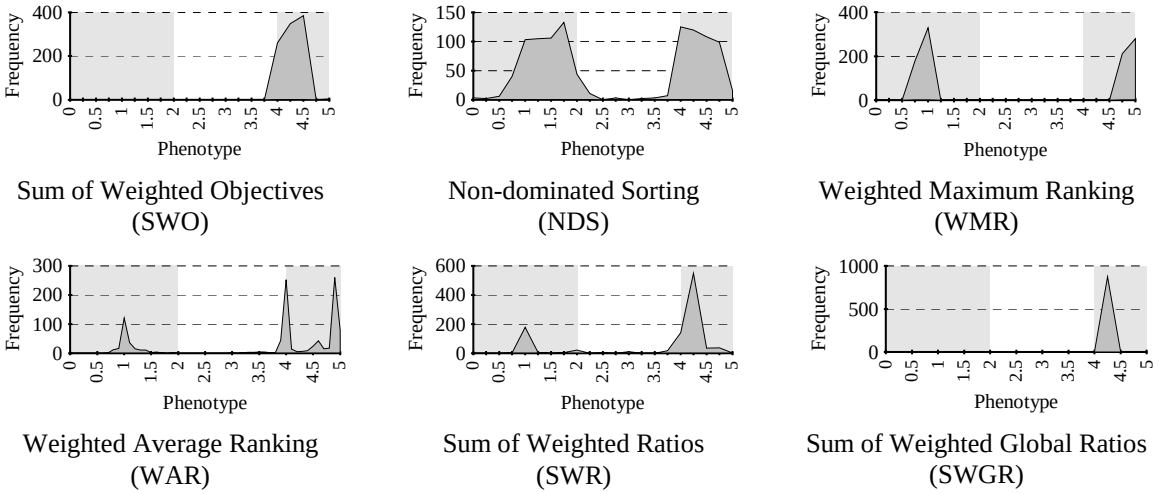


Figure 5. Distributions of solutions within the P-O ranges (shown by grey shaded regions) for function F_3.

6.5. Evolved Results: F_4

Finally, experiments were performed using F_4 with each method in turn. Again, consistent distributions of solutions were obtained, see fig. 6. It should be noted that F_4 is a significant type of function because solutions between the optima of the two objectives are worse than at one optima or the other. This results in two equal best compromise solutions, one at each optima. Hence, although SWO and SWGR this time showed two peaks of distribution, these lie on the best compromise solutions, just as before. Once again, SWR favoured the best compromise solutions with a slightly larger 'spread'. As before, WMR favoured the two optima of the functions with nothing in between. NDS again produced a distribution of solutions covering the entire P-O range, but for this function an unexpected and unwelcome bias towards the middle of the range was evident (where most solutions are very poor). Finally, WAR showed its typically unusual distribution, again favouring values close to the optima of the objectives (and the best compromise solutions, as they are the same as the optima for F_4), with other Pareto-optimal values being favoured less.

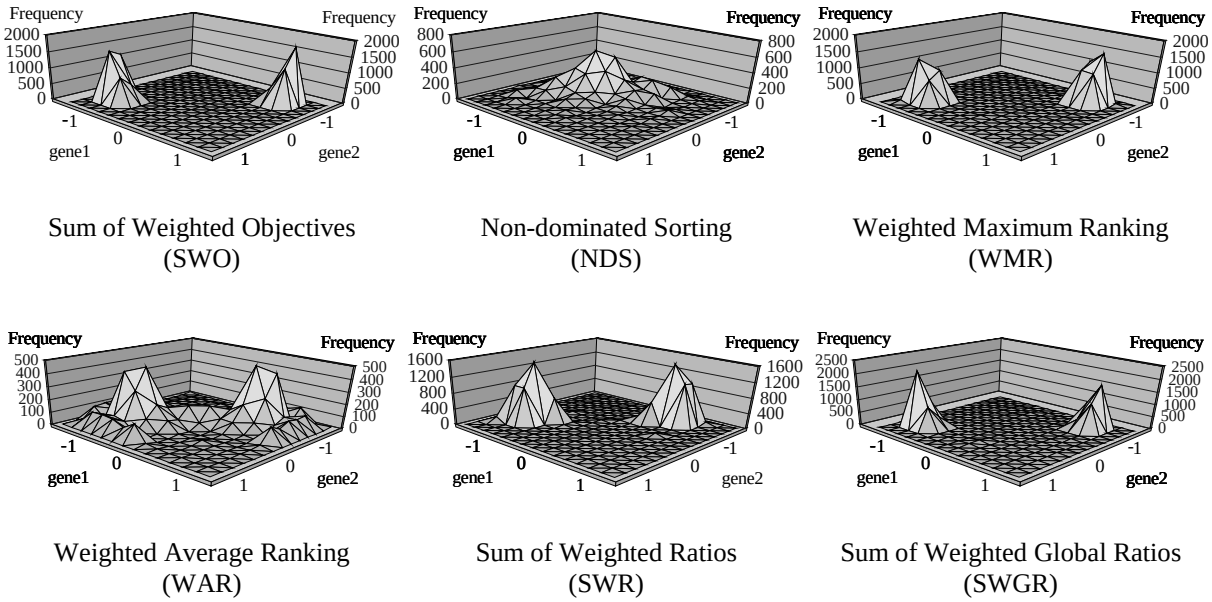


Figure 6. Distributions of solutions within the Pareto-optimal range for function F_4.

6.6. Assessing the Distributions

It should be stressed that all six of the ranking methods allow a GA to produce almost nothing but Pareto-optimal solutions. It is clear, however, that the distribution of these solutions within the Pareto-optimal range is a highly significant factor in determining whether an acceptable solution will be produced.

As the results of the tests show, each ranking method consistently seems to favour certain types of P-O solution, based upon three factors: the Pareto range(s), the separate optimum or optima for each objective and the best compromise solution(s) of the function. These patterns of distributions remain consistent even with more unusual functions with multiple Pareto-ranges (F_3) and multiple best compromise solutions (F_4).

Upon consideration, these distributions are explicable. The three aggregation-based ranking methods: SWO, SWR and SWGR must inevitably favour the best compromise solution(s) to a problem, by definition. (The best compromise solution is the solution with *sum* of weighted objectives minimized, so any ranking method that sums objectives in any way, should have a convergence related to the best compromise value.) NDS gives all non-dominated solutions equal rank, so a fairly even distribution throughout the P-O range is to be expected. WMR bases the fitness of a solution on the maximum rank the solution has for any single objective, so this predictably will result in the generation of solutions only at the optimal of one objective, with nothing in between (a high rank equates to a good value for that objective). Finally, even the unexpected distributions of WAR are explicable. WAR bases the fitness of a solution on the average rank for every objective. This means that a solution with a very high rank for one objective and a low rank for another will be judged equally fit compared to a solution with 'middling' ranks for two objectives. In other words, solutions close to optima of objective functions will be favoured, as will solutions close to the best compromise solution(s).

The results show that NDS, WMR and the new WAR method all give potentially useful distributions of solutions for applications where multiple solutions are required, with predictable, but immovable biases. In contrast, SWO forces the GA to converge on a single solution as close to the best compromise solution as possible, and does allow this bias of P-O solutions to be altered by the user. Unfortunately, because it is range-dependent, its weights must be laboriously set by trial and error in order to define the location of the best-compromise solution(s) in the Pareto range. However, the methods SWR and SWGR both generate solutions in the vicinity of the best-compromise solution(s), and being range-independent, they allow the location of this bias to be easily defined by specifying relative importance values for the objectives. In other words, these two methods allow the location of a subset of acceptable solutions in the P-O range to be defined by specifying which objectives of the problem are more important. The size of the subset depends on which method is used. Hence, for problems where a range of acceptable solutions are desired, biased in favour of those objectives with increased importance, SWR is a suitable choice. For problems where a smaller range, or a even single acceptable solution is desired, SWGR is a suitable choice.

7. Conclusions

This paper investigated the problem of using a genetic algorithm to converge on a small, user-defined subset of *acceptable* solutions to multiobjective problems, in the Pareto-optimal range.

Multiobjective fitness functions cause problems within GAs because the separate objectives have unequal *effective ranges.* If the multiobjective ranking method is not *range-independent*, then one or more objectives in the problem can dominate the others, resulting in evolution to poor solutions.

Because range-independent ranking methods are independent of the problem, they require no weights to fine-tune in order to allow them to rank solutions appropriately into order of overall fitness for a GA. This is a significant advantage over range-dependent methods [1], allowing the same multiobjective GA to be used, unchanged, for a number of different multiobjective problems. Consequently, it would seem that range-independent ranking methods are the most appropriate type of ranking method to use in a general-purpose multiobjective GA.

The concept of *importance* introduced in this paper, allows the GA to converge on a smaller subset of acceptable P-O solutions. Giving certain objectives in a problem greater importance allows ranking methods to generate not just non-dominated solutions, but smaller subsets of acceptable non-dominated solutions at user-defined locations in the Pareto front.

The significance of range-independence and importance in multiobjective ranking methods was shown by the distributions of solutions generated by six methods, applied to four established test functions. The only range-dependent method, SWO, was found to be incapable of coping with objectives with incompatible effective

ranges. The three range-independent methods: NDS, WMR and WAR all produced consistent, and sometimes unusual distributions of P-O solutions, making them potentially useful for some problems. However, only the two new range-independent methods that supported importance: SWO and SWGR, had useful distributions and allowed the bias of their distributions to be easily alterable. Indeed, because of these results, SWGR was chosen to be used within a generic evolutionary design system, which has since been used to tackle a wide range of different solid object design problems (involving the minimization of numerous different multiobjective functions) with great success [1].

References

[1] Bentley, P. J., 1996, *Generic Evolutionary Design of Solid Objects using a Genetic Algorithm*. Ph.D. Thesis, University of Huddersfield, Huddersfield, UK.

[2] Dowsland, K. A., 1995, Simulated Annealing Solutions for Multi-Objective Scheduling and Timetabling. *Applied Decision Technologies* (ADT '95), London, **205-219**.

[3] Fonseca, C. M, & Fleming, P. J., 1995a,. An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evolutionary Computation*, **3:1**, **1-16**.

[4] Fonseca, C. M, & Fleming, P. J., 1995b, Multiobjective Genetic Algorithms Made Easy: Selection, Sharing and Mating Restriction. *Genetic Algorithms in Engineering Systems: Innovations and Applications*, Sheffield, **45-52**.

[5] Linkens, D. A. & Nyongesa, H. O., 1993. A Distributed Genetic Algorithm for Multivariable Fuzzy Control. *IEE Colloquium on Genetic Algorithms for Control Systems Engineering*, Digest No. 199/130, **9/1 - 9/3**.

[6] Marett, R. & Wright, M., 1995, The Value of Distorting Subcosts When Using Neighbourhood Search Techniques for Multi-objective Combinatorial Problems. *Applied Decision Technologies*, London, **189-202**.

[7] Goldberg, D. E., 1989, *Genetic Algorithms in Search, Optimization & Machine Learning.* Addison-Wesley.

[8] Holland, J. H., 1992, Genetic Algorithms. *Scientific American*, **66-72**.

[9] Horn, J. & Nafpliotis, N., 1993, Multiobjective Optimisation Using the Niched Pareto Genetic Algorithm. *Illinois Genetic Algorithms Laboratory (IlliGAL)*, report no. 93005.

[10] Ryan, C., 1994, Pygmies and Civil Servants. *Advances in Genetic Programming*, MIT Press.

[11] Schaffer, J. D., 1984, *Some experiments in machine learning using vector evaluated genetic algorithms*. PhD dissertation, Vanderbilt University, Nashville, USA.

[12] Schaffer, J. D., 1985, Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. *Genetic Algorithms and Their Applications: Proceedings of the First International Conference on Genetic Algorithms*, **93-100**.

[13] Srinivas, N. & Deb, K., 1995, Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, **2:3**, **221-248**.

[14] Sun, Y. & Wang, Z., 1992, Interactive Algorithm of Large Scale Multiobjective 0-1 Linear Programming. *Sixth IFAC/IFORS/IMACS Symposium on Large Scale Systems, Theory and Applications*, **83-86**.

[15] Syswerda, G. & Palmucci, J., 1991, The Application of Genetic Algorithms to Resource Scheduling. *Genetic Algorithms: Proceedings of the Fourth International Conference*, Morgan Kaufmann, **502-508**.

Appendix: Multiobjective Ranking Algorithms

Method 1: Sum of Weighted Objectives (SWO)

```
FOR EVERY SOLUTION IN THE POPULATION
        SOLUTION_FITNESS  =  0
        FOR EVERY CRITERION FITNESS_VALUE 'N' OF THE SOLUTION
                SOLUTION_FITNESS = SOLUTION_FITNESS + WEIGHT [N] * FITNESS_VALUE[N]
```

where WEIGHT [] is an array of pre-defined weights.

Method 2: Non-Dominated Sorting (NDS)

```
CURRENT_RANK = 1
SET EVERY SOLUTION_RANK = UNRANKED
REPEAT
        FOR EVERY SOLUTION IN THE POPULATION
                IF SOLUTION_RANK = UNRANKED AND
                   SOLUTION IS NOT DOMINATED BY ANY UNRANKED SOLUTIONS
                        SOLUTION_RANK = CURRENT_RANK

        CURRENT_RANK = CURRENT_RANK + 1
UNTIL ALL SOLUTIONS HAVE BEEN RANKED
```

Method 3: Weighted Maximum Ranking (WMR)

```
FOR EVERY OBJECTIVE IN PROBLEM
        FORM A LIST OF THE FITNESS OF EACH SOLUTION AND POINTER TO THIS SOLUTION
                FOR CURRENT OBJECTIVE
        SORT FITNESS_LIST INTO ORDER OF FITNESS

SET EVERY SOLUTION.RANK = 99999

FOR EVERY RANKING POSITION 'P' IN POPULATION
        FOR EVERY OBJECTIVE 'O' IN PROBLEM
                IF (FITNESS_LIST FOR 'O' [P] -> SOLUTION.RANK > P / IMPORTANCE [O]
                THEN FITNESS_LIST FOR 'O' [P] -> SOLUTION.RANK = P / IMPORTANCE [O]
```

where IMPORTANCE [] is an array of pre-defined 'importance' weights.

Method 4: Weighted Average Ranking (WAR)

```
FOR EVERY OBJECTIVE IN PROBLEM
        FORM A LIST OF THE FITNESS OF EACH SOLUTION AND POINTER TO THIS SOLUTION
                FOR CURRENT OBJECTIVE
        SORT FITNESS_LIST INTO ORDER OF FITNESS

SET EVERY SOLUTION.RANK = 0

FOR EVERY RANKING POSITION 'P' IN POPULATION
        FOR EVERY OBJECTIVE 'O' IN PROBLEM
                FITNESS_LIST FOR 'O' [P] -> SOLUTION.RANK += P * IMPORTANCE [O]
```

where IMPORTANCE [] is an array of pre-defined 'importance' weights.

Method 5: Sum of Weighted Ratios (SWR)

```
FOR EVERY OBJECTIVE 'O' IN THE PROBLEM
        MAX_FITNESS [O] = WORST FITNESS_VALUE IN CURRENT POPULATION
        MIN_FITNESS [O] = BEST FITNESS VALUES IN CURRENT POPULATION

FOR EVERY SOLUTION IN THE POPULATION
        FOR EVERY CRITERION FITNESS_VALUE 'N' OF THE SOLUTION
                FITNESS_VALUE [N] = (FITNESS_VALUE [N] - MIN_FITNESS [N]) /
                                        (MAX_FITNESS [N] - MIN_FITNESS [N])
FOR EVERY SOLUTION IN THE POPULATION
        SOLUTION_FITNESS  =  0
        FOR EVERY CRITERION FITNESS_VALUE 'N' OF THE SOLUTION
                SOLUTION_FITNESS += IMPORTANCE [N] * FITNESS_VALUE [N]
```

where IMPORTANCE [] is an array of pre-defined 'importance' weights.

Method 6: Sum of Weighted Global Ratios (SWGR)

```
FOR EVERY OBJECTIVE 'O' IN THE PROBLEM
        MAX_FITNESS [O] = WORST FITNESS_VALUE EVER, OF ALL POPULATIONS
        MIN_FITNESS [O] = BEST FITNESS VALUES EVER, OF ALL POPULATIONS

FOR EVERY SOLUTION IN THE POPULATION
        FOR EVERY CRITERION FITNESS_VALUE 'N' OF THE SOLUTION
                FITNESS_VALUE [N] = (FITNESS_VALUE [N] - MIN_FITNESS [N]) /
                                        (MAX_FITNESS [N] - MIN_FITNESS [N])

FOR EVERY SOLUTION IN THE POPULATION
        SOLUTION_FITNESS  =  0
        FOR EVERY CRITERION FITNESS_VALUE 'N' OF THE SOLUTION
                SOLUTION_FITNESS += IMPORTANCE [N] * FITNESS_VALUE [N]
```

where IMPORTANCE [] is an array of pre-defined 'importance' weights.