CHAPTER 5

MULTIPLE OBJECTIVE GROUPING GENETIC ALGORITHM

New opinions are always suspected, and usually opposed,
without any other reason but because they are not already common.

It is one thing to show a man that he is in error,
and another to put him in possession of truth.

-John Locke, an Essay concerning Human Understanding

Keywords: design, genetic algorithm, multiple objective.

1. Introduction

In most real-world problems, we often want to optimise more than one measure of performance at once. Generally, the measures are in conflict with each other, and it can be unsatisfactory to combine them into a single optimisation objective, or reduce them in some way so that only one of them is optimised. A manufacturer-designer asked to design a new assembly line in a short time with a given budget and a specified capacity has to make several major decisions. The trade-off (compromise) between easiness, social issues, reliability, functionality, cost, etc. constitute a set of objectives or goals to reach.

The designer, once having committed to a set of design objectives, and a solution space and methods (tools to search for solutions), has to make a second set of decisions. There are at least two main aspects in which decisions of the designer play a key role in the design process and consequently, in its outcome. First of all, the

designer plays a central role in the specification (the input constraints). Secondly he interprets results. This asks for decisions involving choices among alternatives as well as trade-off among the multiple objective. Indeed, it is the designer who makes use of the objectives during the design process, and hence consciously or unconsciously interprets them. The designer also decides on the solution space to be explored (and the operators to explore it). The decisions made at the different design phases have deep consequences on the outcome of the design process.

Multiple objective optimisation problems in general involve two 'quasi-inseparable' difficulties, namely search and multi-criteria decision making. The space to be searched can be too large to be enumerated, and too complex to be explored by simple search methods. In addition to search space complexity, the multiple objective to be achieved may be conflicting, so that difficult trade-off must be made by a rational decision maker (DM) when he ranks potential solutions.

In general, designers tend (if it is possible) to formulate design problems as mathematical models with many independent parameters that result in one so-called fitness function. This fitness function describes the '*quality*' of the model (or solution) for that set of parameters. It is supposed to model all kinds of conflicting design *demands*. Classical methods tend to aggregate all the objectives in a quite complicated function. The optimisation is then tried on the obtained function. Multiple objective optimisation problems (also called multi-criteria optimisation, multiple performance or vector optimisation) can be defined as the problem of finding a vector of decision variables which optimises a vector function whose elements represent the objective functions. These functions form a mathematical description of performance criteria which are usually in conflict with each other. Hence, the term 'optimise' means finding a solution which would give to the designer values acceptable for all the involved objective functions.

In a more general setting and especially when we have to design an artifact, we have to deal with two kinds of user's *preferences*. The first kind of preferences concerns the content of each obtained design. The second kind of preferences deals with a set of objectives (goals) and arises each time we have to decide about the best solution among a set of valid solutions–all more or less satisfying the first kind of preferences. This second problem has to deal more with how the DM judges a set of solutions. This might involve assigning different utilities (or preferences) to different objectives and combining them into some figure of merit. The difficulty with the specification of one compromise decision lies in the assessment of weights of the aggregate utility function which reflects the parties power, and intensity.

The most general orientation in the search of methods to solve problems consists in deriving exact mathematical results. However, such exact solutions may not be always available, or may be too non-general to be of much use. Generally, the primary orientation is towards deriving concise and precise estimates of quantities of interest. The accent is put after on approximating solutions to problems. A problem carries a notion of *size* and *convergence* and we are interested in approximations that become more accurate as the size becomes large. Even if it is difficult to analyse the

convergence of exact methods on well defined problems, it is quite common to talk about convergence–which means the way and the time needed to reach a best or a good solution of a given problem on a given kind of machines... In contrast, when dealing with multiple objective problems, talking about convergence seems a rather difficult task, since there is no common agreement on what the optimum really means.

In general, due to the design difficulties and the human limits, two designers will frequently come up with solutions that logically make sense, but are different in some manner. This difference is due to the way the two designers judge the proposed solution of their design problem. The difference between two proposed designs depends on the user's preferences. Thereafter, the quality of a given solution is designer-dependent.

The remainder of this chapter is organised into 6 sections. Our point of view on multiple objective problems is uphold in section 2. In section 3 we briefly review related work on multiple objective problems. Section 4 describes the grouping genetic algorithms while section 5 presents its adaptation to multiple objectives problems. Section 6 is devoted to an academic case study. Finally we draw conclusions and present some applications of the presented method in section 7.

2. Multiple objective optimisation

In single objective optimisation problems, the feasible set is totally ordered according to the objective function *f*. For two solutions s_1 and s_2 one have either $f(s_1) > f(s_2)$ or $f(s_1) \leq f(s_2)$. In contrast, multiple objective problems (MOP) present a set of *optimal* solutions which are quite difficult to order. The solutions once evaluated produce a vector whose components represent a trade-off in the *decision* search space. A DM then implicitly chooses an *acceptable* solution by selecting one of these vectors.

The concept of Pareto optimum was formulated by Vilfredo Pareto in 1896 and constitutes by itself the origin of research in multiple objective optimisation (Pareto, 1988). A solution S_1 is Pareto optimal if there exists no feasible vector S_2 which would decrease some criterion without causing a simultaneous increase in at least one criterion.

In most design problems there are always *restrictions* imposed by the particular characteristics of the environment or resources available. These restrictions must be satisfied in order to consider that a certain solution is acceptable. In general, all these restrictions are called *constraints*, which describe dependencies among decision variables and parameters involved in the problem. These constraints are frequently expressed in form of mathematical inequalities.

Most real-world engineering design problems are multiple objective by nature, since they normally present several (possibly conflicting) objectives that must (*if possible*) be satisfied at the same time. The word 'optimum' has several interpretations within this

context, and it is up to the designer to decide which solution fits better to his/her application (*design goals*). Currently, there are many (more than 10) mathematical multiple objective optimisation techniques, differing by the way they understand the term 'optimum'.

In the case of multiple objective problems, instead of a single 'optimal' solution the competing goals give rise to a set of *compromise* solutions. In the absence of *preference* information, none of the corresponding trade-off can be said to be better than the others. On the other hand, the search space can be too large and too complex to solve the problem by exact methods. Thus, efficient *optimisation* methods such as the genetic algorithms are required.

As discussed above, since we are missing the notion of *optimum*[1] it is quite misleading to talk about fitness. Indeed, the fitness or the quality of a given solution is difficult to evaluate–it depends on the values attributed to the different objectives of the given solution. It is more safe to talk about ranking of solutions rather than fitness. Talking about ranking gives rise to decision making problem which is quite popular among deciders community and quite less among the optimisation community until few years.

Search and decisions are not independent tasks. Making some *choices before search* can reduce the size of the search space, while *search before decision making* can eliminate a vast number of *desired* solutions and focus decision making on a few alternatives. The third approach is situated in the middle of the two cited approaches, it is an *integration of a search and decision*. The type and the degree of the integration define the strength of the *exploration* and the *exploitation*.

The two classic (pragmatic) strategies that were applied with the traditional *separation* of search and multi-criteria decision-making can be described as follows.

- *First, make multi-criteria decision to aggregate objectives, then apply a search method to optimise the resulting figure of merit.* The different objectives are combined to form a scalar objective function, usually through a linear combination (weighted sum) of the attributes. The weights estimate the importance of each objective. This approach is very simple and easy to implement. The weighting coefficients represent the relative importance of the objectives. This means that the problem is transformed into a scalar optimisation one. The approach is well suited to proportional non competing objectives (an improvement of an objective leads to an improvement or indifference of the others). Indeed, if there exists a solution that simultaneously succeeds in minimising (or maximising) each of the different objectives, this approach can be reasonably satisfactory. Because, optimise an objective will also optimise the others. Unfortunately, this is generally not the case of real-world problems. The main drawback of this approach is the fact that it may use the sum of values of two totally different objectives (for instance it

[1] The word 'optimise' means finding such a solution which would give the values of all the objective functions acceptable to the DM.

could sum the cost value with the reliability value), which makes no sense. The second problem with this approach is precisely how to determine such weights when there is not enough information about the problem.

- *Conduct the search using the different objectives at the same level of importance.* In the more general case, the objectives compete, in the sense that an improvement of a given objective will in some cases lead to a degradation of others. In this case, the approach of forming a weighted sum is less attractive, because the choice of weights will determine the trade-off between the various component objectives that optima of the combined function will exhibit. This is particularly unsatisfactory in cases where the various objectives are non-commensurate, in the sense that trade-off between them are either arbitrary or meaningless. An example of this is cost an reliability. The aim is to increase the reliability and to decrease the cost. Increasing one objective (as instance cost) increases the other one (reliability), which is not the global aim. In the case of such multiple objective problems, a more satisfactory approach is to search not for a single solution but for that set of solutions that represent the 'best possible tradeoffs'. This leads to a set of alternative solutions and the search phase is followed by making multi-criteria decision to choose among the reduced set. This approach yields the Pareto frontier. Referring to Figure 5.1, *O* is unique among *A*, *B*, *C*, and *D*: its corresponding decision vector $O=(o_1, o_2)$ is not dominated by any other decision vector. That means, *O* is optimal in the sense that it cannot by improved in any objective without causing a degradation in at least another. Such solutions are denoted as Pareto optimal (non-inferior). The Pareto optimisation aims to provide the DM with a representative set of solutions from the Pareto optimal front, so that he can see the actual tradeoffs that have to be made in choosing a solution. The search phase is then followed by making multi-criteria decision to choose among the reduced set. This approach is generally considered to represent a 'best practice'. The problem is the number of solutions the DM has to choose among them. The human cannot easily decide among more than few solutions.

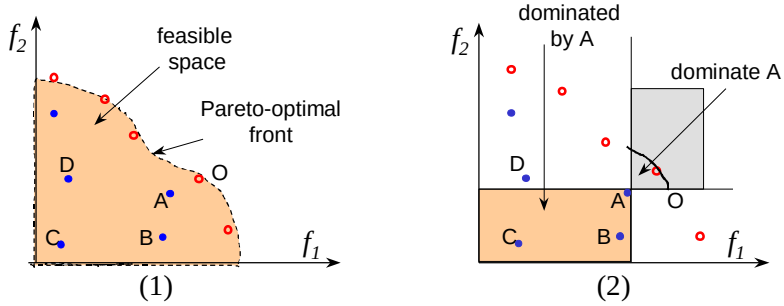

Figure 5.1. Pareto optimality (1) and the dominance relations in objective space (2).

Our novel approach is to integrate the decision and the search and permits to deal with user's preferences. In the absence of preference information, none of the corresponding tradeoffs of the Pareto-frontier can be said to be better than the other

ones. Thus, decision (using preferences) and search are merged and interact mutually. More details about the method can be found in section 5. Evolutionary methods, particularly GAs, possess several characteristics that are desirable for MOPs and make them preferable to classical optimisation methods. GAs have been viewed to be, since their early days, well suited for MOPs. Indeed, various evolutionary approaches to MOPs have been proposed since 1985 (Deb, 1999). Some expert systems were developed, intending to help the user to find the most appropriate multiple objective optimisation technique according to the characteristics of the problem to be solved and the available computer resources. In the next section, the background to our work and particularly to the evolutionary methods is described.

3. State of the art

Many authors indicate that the notion of genetic search in a MOP dating back to the late 60s. Rosenberg's (Rosenberg, 1967) study contained a suggestion that would have led to multi-criteria optimisation if he had carried it out as presented. In general, the GAs require scalar fitness information to work, which means that when approaching MOPs, one needs to transform the problem at hand into a single objective problem. One problem is that it is not always possible to derive a global criterion based on the formulation of the problem. In the absence of information, objectives tend to be given equivalent importance, and when we have some understanding of the problem, we can combine them according to the information available, probably assigning more importance to some objectives. Optimising a combination of objectives has the advantage of producing a single compromise solution, requiring no further interaction with the decision maker.

While covering the existing literature it seems that the main difference among the cited methods is the way solutions are *ranked*. There exist three ranking methods: the *aggregating* approaches, the *non-Pareto* approaches and the *Pareto* approaches. Three other derivations are the *local search* approaches, methods dealing with *preferences* and methods dealing with *constrained search spaces*.

3.2. Use of aggregating functions

Several attempts such as *aggregation* are reported in the literature. Different techniques have been made to combine the objective function in different ways.

Weighted sum approach

Historically, multiple objective have been combined to form a scalar objective function (Fonseca, 1995). The weights express the importance of each objective. Generally, a simple linear combination of objectives is used. The problem with this approach is precisely how to determine such weights when there is not enough information about the problem. Indeed, the method is very subjective, and may neglect the importance of some objectives, and it is often hard to find weights which

can accurately reflect the real situation. In general, the obtained 'optimal' point is a function of the coefficients used to combine the objectives.

Reduction to a single objective

Another way to deal with MOPs is to optimise with regard to one objective, the other objectives remaining constants (constrained to a single value) (Ritzel, 1994). Then, through a process of running the method numerous times with different values of the constrained objectives, a tradeoff surface can be constructed. The main drawback of this approach is that it is time-consuming, and the coding of the objective functions may be difficult or even impossible for certain problems. The method cannot deal with designer's preferences, since the optimisation is run each time for a single objective without any preference.

Goal attainment

In this method, a vector of weights relating the relative *under-* or *over-attainment* of the desired goals must be elicited from the DM in addition to the goal vector (Wilson, 1993). The DM has to assign targets or goals that he wishes to achieve for each objective. These values are incorporated into the problem as additional constraints. The objective function then try to minimise the absolute deviations from the targets to the objectives. By varying the weights, it can generate the set of non-dominated solutions. In the case of under-attainment of desired goals, a smaller weighting coefficient is associated with a more important objective. For over-attainment of desired goals, a smaller coefficient is associated with a less important objective.

Use of penalty functions

This method is based on both 'constraints satisfaction' method and 'weighting objectives' method. Once more, the assignment of *penalty* values is not always a trivial task. This method normally produces only the *min-max* optimum, but not the Pareto optimum or the Pareto front. The basic idea is to 'punish' the fitness value of a solution whenever the produced solution violates some constraints. Theoretically, the penalty decreases when the value of the penalty function coefficient is increased and *convergence* is achieved by increasing the penalty function coefficient to infinity. However, a large value for the penalty function coefficient causes bad conditioning in the optimisation process and results in numerical instability or slow convergence. Furthermore, since the value of the penalty function coefficient is unknown, much experimentation is required to find an appropriate value. The penalty functions are generally problem dependent, and therefore difficult to establish.

3.3. Non-Pareto approaches

To overcome difficulties as well as the limitations involved in the aggregating approaches, work has been devoted to development of alternative approaches based on ranking. Next are examined some of the most popular non-Pareto approaches.

VEGA

Schaffer (Schaffer, 1985) was probably the first to recognise the possibility of exploiting Evolutionary Algorithms to treat MOPs. His approach was to use an extension of the simple GA (called vector evaluated GA 'VEGA'), and that differed from the standard one in the way the selection was performed. At each generation a number of sub-populations were generated by performing proportional selection according to each objective function in turn. He used a special selection mechanism which chooses *k* equally sized subgroups of individuals from the population based on their performance in each of the *k* criteria. These sub-populations are then shuffled together to obtain a new population. It was recognised that this would favour solutions with extreme performance in at least one objective. Schaffer suggested applying fitness penalties to locally dominated points and redistributing the deducted fitnesses to non-dominated ones. This caused a premature convergence because in populations with few non-dominated points, these points were given large fitness values. Later analysis of VEGAs performance showed that the fitness was a linear combination of criteria, the weights (in this linear combination) being defined by the distribution of the population. Due to the nature of the used selection, the population contained many extreme individuals and few compromise ones.

Lexicographic ordering

Fourman (Fourman, 1985) introduced a lexicographic ordering, where the basic idea is that the designer ranks the objectives in order of importance. The optimum solution is then found by minimising the objective functions, starting with the most important one and proceeding according to the order of importance of the objectives. In a first version of his algorithm, objectives were assigned different priorities by the user and each pair of individuals were compared according to the objective with the highest priority. If this resulted in a tie, the objective with the second highest priority was used, and so on. A second version of this algorithm, consisted of randomly selecting the objective to be used in each comparison. As in VEGA (Schaffer, 1985), this corresponds to averaging fitness across fitness components, each component being weighted by the probability of each objective being chosen to decide at each tournament. However, the use of in-twos comparisons makes an important difference with respect to VEGA, since in this case scale information is ignored.

Evolutionary Strategies

Kursawe (Kursawe, 1991) formulated a multiple objective version of evolutionary strategies (ES) (Schwefel, 1981). Selection consisted of as many steps as objective functions. At each step, one of these objectives was selected randomly according to a probability vector, and used to delete a fraction of the current population. After selection, the survivors became parents of the next generation. The map of the tradeoff surface was produced from the points evaluated during the run. Since environment was allowed to change over time, diploid individuals (dominant and recessive) were necessary to keep recessive information stored.

Weighted sum

Hajela and Lin (Hajela, 1992) included the weights of each objective in the chromosome and promoted their diversity in the population through fitness sharing. Their goal was to be able to simultaneously generate a family of Pareto optimal solutions corresponding to different weighting coefficients in a single run of the GA. Besides using sharing, the authors used a vector evaluated approach based on 'VEGA' to achieve their goal. The approach belongs to the category of aggregation selection with parameter variation. The diversity of the weight combinations is promoted by fitness sharing in the objective space. As a consequence, the method evolves solutions and weight combinations simultaneously. The weighted-sum aggregation appears still to be widespread due to its simplicity.

3.4. Pareto-based approaches

While in single objective optimisation the optimal solution is usually clearly defined, this does not hold for MOPs. Instead of a single optimum, there is rather a set of alternative tradeoffs, generally known as Pareto-optimal solutions. These solutions are optimal in the wider sense that no other solutions are superior to them when all objectives are considered. In the following, we will review some of the main Pareto-based approaches.

Pareto-based fitness assignment

One of the first attempt to resolve the limitations of Schaffer's approach was proposed by Goldberg (Goldberg, 1989). He suggested the use of non-domination *ranking* and *selection* to move a population toward the Pareto front in a MOP. The idea is to find the set of solutions in the given population that are Pareto *non-dominated* by the rest of the population. These solutions are then assigned the highest rank and eliminated from further contention. Another set of Pareto non-dominated solutions are determined from the remaining population and are assigned the next highest rank. This process continues until the population is suitably ranked. The author also suggested the use of some kind of *niching* to keep the GA from converging to a single point on the front. A niching mechanism would allow the GA to maintain individuals all along the non-dominated frontier.

Multiple objective genetic algorithm

Fonseca and Fleming (Fonseca, 1995) proposed a scheme in which the rank of a certain individual corresponds to the number of chromosomes in the current population by which it is dominated. All non-dominated solutions are assigned rank *1*, while dominated ones are penalised according to the population *density* of the corresponding region of the tradeoff surface. This type of *blocked* fitness assignment is likely to produce a large *selection pressure* that might produce premature convergence. To avoid that, the authors use a *niche-formation* method to distribute the population over the Pareto-optimal region. The basic concept of the method has been extended

meanwhile by the adaptive fitness sharing and continuous introduction of random immigrants (Fonseca, 1998).

Non-dominated sorting genetic algorithm

Srinivas and Deb (Srinivas, 1994) proposed the non-dominated sorting genetic algorithm (NSGA). The method is based on several layers of classifications of the individuals. Before the selection is performed, the population is ranked on the basis of non-domination: all non-dominated individuals are classified into one category. Each solution is assigned a *dummy* fitness value–this fitness is proportional to the population size. In order to maintain diversity of the population, these classified individuals are shared with their dummy fitness values. This group of classified individuals is then ignored and another layer of non-dominated individuals is considered. The process continues until all individuals in the population are classified. Solutions in the first front have the maximum fitness value, and by the way they get more copies than the rest of population. This allows a quick convergence of the population toward non-dominated regions. The efficiency of the method lies on the way objectives are reduced to a dummy fitness function using a non-dominated sorting procedure. With this approach, any number of objectives can be solved.

Niched Pareto GA

Horn and Nafpliotis (Horn, 1993) used a Pareto domination tournament (NPGA) instead of non-dominated sorting and ranking selection method. Instead of limiting the comparison to two individuals, a number of other individuals was used to help determine dominance. A comparison set comprising a specific number t_{dom_pres} (dominance pressure) of individuals is picked at random from population at the beginning of each selection process. Two random individuals are picked to select a winner in a tournament selection. Both individuals are compared with the members of the comparison set for domination with respect to objective functions. If one of them is non-dominated and the other is dominated, then the non-dominated individual is selected. If both are either non-dominated or dominated, a niche count is found for each individual in the entire population. The niche count is calculated by simply counting the number of points in the population within a certain distance from an individual. The individual with least niche count is selected. When, both competitors were either dominated or non-dominated (i.e., there was a tie), the result of the tournament was decided through fitness sharing (Goldberg, 1989).

3.5. Preferences and local search methods

Local search

Jaszkiewicz (Jaszkiewicz, 1998) presented a random directions multiple objective genetic local search metaheuristic (RD-MOGLS). The method is based on the idea of simultaneous optimisation of all possible utility functions. At each iteration of the method a weighted Tchebycheff or weighted linear utility function is drawn at random. Then, a sample of best solutions is selected from the set of prior generated

solutions. The sample is treated as a temporary genetic population. A number of randomly selected pairs of solutions from the temporary population are recombined.

Multi-objective genetic local search (MOGLS)

Murata et al. (Murata, 1999) proposed a specification method of the local search direction for each solution in the multiple objective genetic local search algorithm (MOGLS). The MOGLS is a GA-based hybrid algorithm for finding a set of Pareto-optimal solutions. They used a weighted sum of multiple objective as a fitness function for selecting a pair of parents solutions. The fitness function is also employed in the local search procedure for a new solution generated from the selected parent solutions. The iterative improvement of the newly generated solution is performed in the multi-dimensional objective space based on the local search direction specified by the weight values in the fitness function. The main feature of the selection procedure is that the weights attached to the multiple objective functions are not constant but randomly specified for each selection. The elite preserve strategy uses multiple elite solutions instead of a single elite solution. That is, a certain number of individuals are selected from a tentative set of Pareto optimal solutions and inherited to the next generation as elite individuals.

Preferences for MOGA

Cvetkovic and Parmee (Cvetkovic, 1999) presented a method based on preference relations transforming qualitative relationships between objectives into quantitative attributes. The preference order method is similar to linguistic methods. For every two objectives, a preference (or an order) must be given using the following characteristics:{*less important, equally important, much more important, much less important, more important*}. The qualitative preferences between the different objectives yield a *preferences* matrix "R". This matrix permits to compute the weights of the given objectives. The method is integrated in the weighted sum GA and a combination of Pareto and weighted GA. A modified Pareto method for MOP is presented. The method yields a Pareto front dealing with preferences.

3.6. Constrained problems

The presence of 'hard' constraints in a MOP may cause further difficulties. It is sure that the success of a multiple objective GA in tackling these problems depends on the constraint-handling technique used. Traditionally, a simple penalty-function based method has been used to penalise each objective function. Penalty function methods demand an appropriate choice of a penalty parameter of each constraint. Many other techniques are reported in the literature.

Multi-objective evolution strategy for constrained optimisation problems

To (To, 1997) presented a MultiOBjective Evolution Strategy method (MOB-ES) for solving MOPs subject to linear and non-linear constraints. The method is based on

the concept of *C*- (constraints), *F*- (objective function) and *N*- (niche)fitness, which allows to handle constraints and (in)feasible individuals. The author provided new ideas for handling (in)feasible individuals, since some niche infeasible individuals can be better than some feasible ones. The use of niche infeasible individuals enables to avoid an unnecessary concentration of the population at local optima and quickly to shift the population towards the feasible global optima.

Strength Pareto evolutionary algorithm

Ziztler (Ziztler, 1999) introduced an evolutionary approach to multi-criteria optimisation, called the Strength Pareto EA (SPEA). The method *combines several features of previous* multiple objective EAs in a unique manner. Like other MOEAs, the method stores non-dominated solutions in an external and continuously updated population. The individual's fitness depends on the number of non-dominated points (in external population) that dominate it. The population diversity is preserved using the Pareto dominance relationship. The method performs a clustering procedure to reduce the number of individuals externally stored without destroying the characteristics of the tradeoff front.

Constraints handling through a multiobjective optimisation technique

Coello (Coello, 1999) proposed a population-based approach similar to VEGA to handle constraints. The proposed approach does not rank individuals, but it uses instead different fitness functions for each of the sub-populations allocated, depending on the feasibility of the individuals contained within each of them. The VEGA approach (Schaffer, 1985) is known to have difficulties in MOPs due to the fact that it tries to find individuals that excel only in one dimension regardless of the others. These drawbacks turns out to be an advantage in the case of constrained problems, because the aim is to find solutions that are completely feasible, instead of good compromises that may not satisfy one of the constraints.

For a comprehensive review, see the overview of different MO-GA methods presented by Fonseca and Fleming (Fonseca, 1995) and Coello (Coello, 1999). Van Veldhuizen (Van Veldhuizen, 1999) made a survey on test problems. Until now, there is no systematic study to speculate what problem features may cause an MO-GA to face difficulties. Deb (Deb, 1999), identified a number of features and suggested a methodology to construct test problems from single-objective optimisation problems. The author believes that more studies are needed to better understand the working principles of a MO-GA.

4. Grouping problems and the grouping genetic algorithm

4.1. Grouping problems

Since our main objective in this study is design of assembly lines which can be simply described as a problem of assignment of tasks to stations. The problem can be easily

transformed to a grouping problem. In the following we will introduce the main features of the grouping genetic algorithm developed by Falkenauer (Falkenauer, 1998).

4.2. Grouping genetic algorithm (GGA)

Falkenauer (Falkenauer, 1998) pointed out the weaknesses of standard GAs when applied to grouping problems, and introduced the grouping genetic algorithm (GGA), which is a GA heavily modified to match the structure of grouping problems. Those are the problems where the aim is to group together members of a set (i.e. find a good partition of the set). The GGA's operators (crossover, mutation and inversion) are group-oriented, in order to follow the structure of grouping problems.

Encoding scheme

The most distinctive feature of the GGA is the special solution *encoding* it uses. Falkenauer (Falkenauer, 1998) indicated several drawbacks of standard GAs applied to grouping problems stemming from the fact that the schemata processed by either the classic GA of Holland (Holland, 1975) or the ordering GA of Goldberg (Goldberg, 1989) do not represent meaningful regularities of the search space of grouping problems.

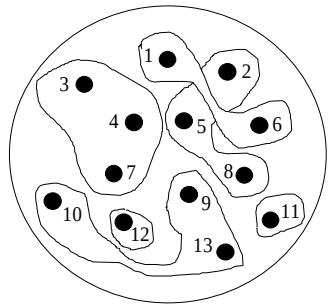

Figure 5.2. A grouping and the corresponding GGA chromosome.

In a grouping problem, the groups constitute the natural regularity of the search space, since the cost function to optimise is defined over the groups rather than isolated objects (members of the set being partitioned). Given the fact that schemata processed by a GA are defined over the genes in the chromosomes, it thus follows that a GA adapted for grouping problems must cast groups as genes in its chromosomes.

Consequently the genes in the GGA's chromosomes encode groups of objects, rather than the objects themselves. Figure 5.2 illustrates the idea. The objects in the

upper part of the figure are grouped into six groups of various sizes. Likewise the corresponding chromosome depicted in the lower part of the figure features six genes, each of them encoding a group of one more objects.

Crossover operator

A crossover's job consists of producing offspring out of two parents in such a way that the children inherit as much as possible of the meaningful information from both parents. Since it is the groups that convey important information in grouping problems, we must find a way to transmit groups from the parents onto the children. The GGA crossover proceeds as follows (see Figure 5.3).

1. Select at random two crossing sites, delimiting the crossing section, in each of the two parents.
2. Inject the contents of the crossing section of the first parent at the first crossing site of the second parent. Recall that this means injecting some of the groups from the first into the second.
3. Eliminate all objects now occurring twice from the groups they were members of in the second parent, so that the 'old' membership of these objects gives way to the membership specified by the 'new' injected groups. Consequently, some of the 'old' groups coming from the second parent are altered: they do not contain all the objects anymore, since some of those objects had to be eliminated.
4. If necessary, adapt the resulting groups, according to hard constraints of the problem and the cost function to optimise. At this stage, local problem dependent heuristics can be applied.
5. Apply the points 2 through 4 to the two parents with their roles reversed in order to generate the second child.

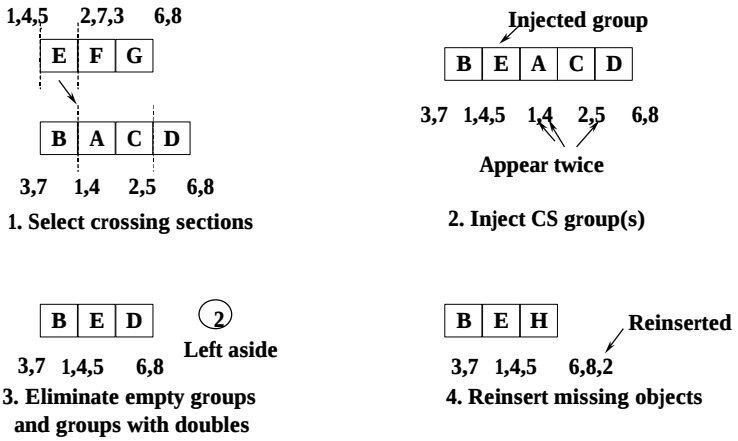

Figure 5.3. The GGA crossover operator.

Note that the child inherits genes from the first parent (its crossing section injected in point 1) as well as from the second (the genes not affected by the elimination in point 3). Since the genes encode groups, the child therefore inherits groups from both parents, as required by the structure of grouping problems.

Mutation operator

According to the nature of the particular grouping problem, one or more of the following three operators can be applied: create new group(s) from randomly selected objects; eliminate a randomly selected group by distributing the objects it contains over the other groups; shuffle a small number of objects among groups.

Inversion operator

The inversion serves to shorten promising schemata made of coadapted genes. In the GGA, the mechanism is the same as the operator of Holland (Holland, 1975), i.e. a segment on the chromosome is selected at random and the order of genes in that segment is inverted.

For more details about the GGA and its applications, the reader is asked to refer to (Falkenauer, 1998).

5. Multiple objective grouping genetic algorithm (MO-GGA)

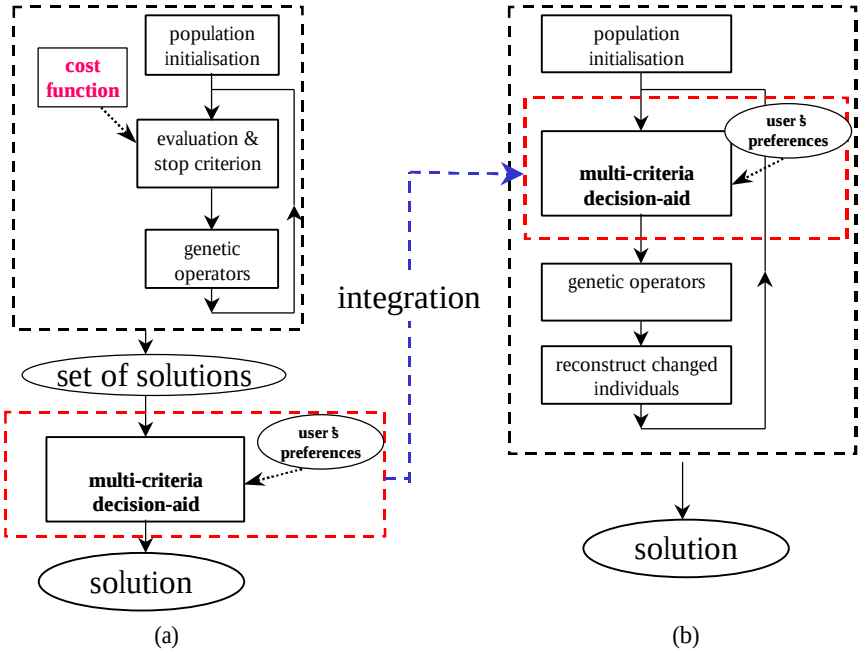

Figure 5.4. Integrating search and decision making into the GA.

Selecting a solution from a set of possible ones on the basis of several objectives can be considered as a difficult and intriguing problem. The main difference between existing methods is the way the 'best' solution is chosen from a set of valid ones. We use a multi-criteria decision-aid (MCDA) method to deal with such problems.

Figure 5.4(a) presents the most used approach to MOPs the GA generates a set of Pareto solutions and the DM uses his preferences to choose the best solution. Thus,

the MCDA is used after the search phase. Making some decision after the search phase can eliminate a vast number of desired solutions and let us focus on a few alternatives. As introduced above, an integration of a search and decision was proposed. The main problem in MOPs is how to evaluate the quality of a given solution and how to measure the closeness of solution to the optimal solution in the search space. Indeed, there is no common agreement on what *optimum really means*. The approach we propose is based on a merge of a search and a MCDA as illustrated in Figure 5.4(b). Indeed, in order to come out of the MOP stated by the *cost function*, the MCDA method called PROMETHEE II[2] is used as a ranking technique. For more details about the PROMETHEE II method, the reader is invited to refer to (Brans, 1994). It is however important to know that it computes a net flow f which is a *kind of fitness* of each solution. This 'fitness' gives a ranking between the different solutions of the population.

The most used methods in that MCDA field are ELECTRE (Roy, 1986) and PROMETHEE. It is obvious that those methods have a lot of similarity because they are based on the same philosophy, which consists in accepting to be less rigorous mathematically in order to be able to fit better with the uncertainty of human reasoning. We have preferred to use PROMETHEE because it is simpler to use than ELECTRE and easier to understand and manipulate by a novice in the field of MCDA.

The PROMETHEE II method is known as one of the most efficient but also one of the easiest in the MCDA field. Given a_1 , a_2 , ... a_i , ... a_n a set of n potential alternatives and f_1 , f_2 , ... f_j , ... f_k k evaluation criteria. Each evaluation $f_j(a_i)$ is a real number. This set of data can be presented in a matrix format as shown in Table 1.

In the case of GA each potential alternative a_i is an individual of the population and the evaluation criteria $f_j(a_i)$ is the value of objective j for individual a_i. The ranking of a given population starts when the evaluation matrix is available. PROMETHEE then computes the 'net flow' f_i for each individual i. Weights (associated to each objective) are involved in the computation of the f number and represents the relative influence of each objective. Thus, solutions are not compared according to a cost function yielding an absolute fitness of individuals as in a classical GA, but are compared to each other thanks to the flows, depending on the current population.

	$f_1(.)$	$f_2(.)$	...	$f_j(.)$	...	$f_k(.)$
a_1	$f_1(a_1)$	$f_2(a_1)$	...	$f_j(a_1)$	...	$f_k(a_1)$
a_2	$f_1(a_2)$	$f_2(a_2)$	...	$f_j(a_2)$	...	$f_k(a_2)$
...						
a_i	$f_1(a_i)$	$f_2(a_i)$	...	$f_j(a_i)$	...	$f_k(a_i)$
...						
a_n	$f_1(a_n)$	$f_2(a_n)$	...	$f_j(a_n)$	...	$f_k(a_n)$
	ϕ_1	ϕ_2		ϕ_j		ϕ_k

Table 1. PROMETHEE II evaluation matrix.

[2] A short description of the PROMETHEE II method is given in Appendix 3.

At each generation, a ranking change the fitness of individuals with their *environment* (the current individuals in the population). In classical GAs, the fitness of an individual is independent of the other individuals constituting the population. There is no direct feedback from the environment to the individual's fitness which remains constant, *unaffected* by their environment. We believe this is an *handicap* of a GA-based methods in the case of MOPs.

The values of the f are context related and have no absolute meaning. Hence, it becomes impossible to fix a stop criterion for the GA. The optimisation is stopped at user's request, or if no better solution has been found for a given number of generations.

5.1. Control strategy

As the fitness of the individuals is content dependent, solutions have to be compared to the best ever found one. At each generation, a set of old individuals, the new individuals as well as the best-ever solution are involved in the evaluation of the whole population (Figure 5.5). Suppose that the values of the different objectives of each individual are separately evaluated in advance. The MCDA method ranks the individuals taking into account the presence of the others. This fitness allows the GA to choose the best solution simply by looking for the individual having the maximum value f.

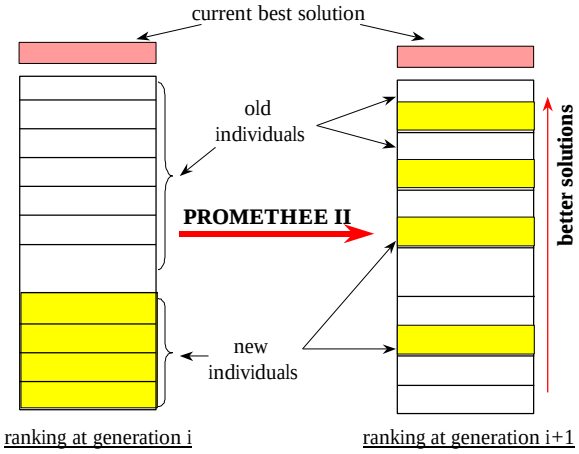

Figure 5.5. Control strategy of the MO-GA.

5.2. Individual Construction Algorithm

Evolutionary algorithms typically seed the initial population with entirely random values (i.e., starting from scratch). Evolution is used to discover which of the randomly sampled areas of search space contain better solutions, and then to converge upon that area. Sometimes, the entire population is constructed from

random mutants of a single (created randomly or user-supplied) solution. Often random values are generated between specified ranges (a form of constraint handling). It is not uncommon for explicit constraints handling to be performed during initialisation, by deleting solutions which do not satisfy the constraints and creating new ones. More complex problems often demand alternative methods of initialisation.

In our opinion, how the initialisation is done is critical as it influence the convergence of the GA. Thus, any explicit knowledge about the system being optimised have to be used to initialise population. However, some kind of randomisation must also be used to create diversity in the population. For a given problem, the Individual Construction Algorithm (ICA) is an algorithm (exact method or a heuristic) used to create individuals (solution to a problem at hand). In the case of ALB problem an ICA called 'equal piles' was proposed (Chapter 6) meanwhile the 'equal piles/branch and cut' ICA was presented for the RP problem (see Chapter 7).

5.3. Overall architecture of the evolutionary method

The choice of one solution over the others requires problem knowledge. It is the DM task to adjust the *weights* to help the algorithm to find good solutions. Optimising a combination of the objectives has the advantage of producing a single solution, requiring no further interaction with the DM. For given user's preferences and a given design problem we run the following multiple objective GA.

The initial population is generated using an ICA just like in the simple GAs. The individuals are then ranked using the MCDA method (PROMETHEE II). At each iteration of the main loop, the better solutions are selected from the current population. Recombination produces a number of new individuals (offspring). The mutation is used to explore the search space. The offspring is then incorporated into the original population. Again, the individuals are ranked using the MCDA. The loop finishes when the termination criteria given by the user (e.g., convergence, maximum number of generations, etc.) are reached.

The basic features of the MO-GGA are illustrated in the following pseudo-code.

Generate an initial population with an individual construction algorithm*;*
Order individuals using PROMETHEE II;
repeat
    Select parents;
    Recombine best parents of the population;
    Mutate children;
    Reconstruct individuals using the ICA;
    Replace individuals of the population by children;
    Use PROMETHEE II to order the new population;
until *a satisfactory solution has been found.*

This hybrid algorithm presents the advantage of allowing a real multi-criteria optimisation since one considers to optimise simultaneously multiple objectives. Note that the weights associated to the different objectives used in PROMETHEE II allow an easy matching of the user's wishes.

5.4. Branching on populations

The presented GA-based method is a merge of search and MCDA. Classically, the idea of *'weight associated to objectives'* is less-loved since it leads to rather artificial function and it is thus difficult to tune the weight of each objective. Because of the variability in the results, inherent to the stochastic nature of GAs, it is common to run GA several times for different preferences and to take the best result. However, it is possible to save the GA's population at some intermediate states and restart from one of these populations rather than from the very beginning. By doing so, we generate a tree of populations, where a child node is generated from its parent by running a number of GA iterations. This is what we call the *branching on population*[3] technique (Figure 5.6). The idea is based on artificial intelligence search techniques like *branching* and *backtracking*. The method is inspired from the work of Steinberg et al. in their optimisation method by searching a tree of populations (Steinberg, 1999).

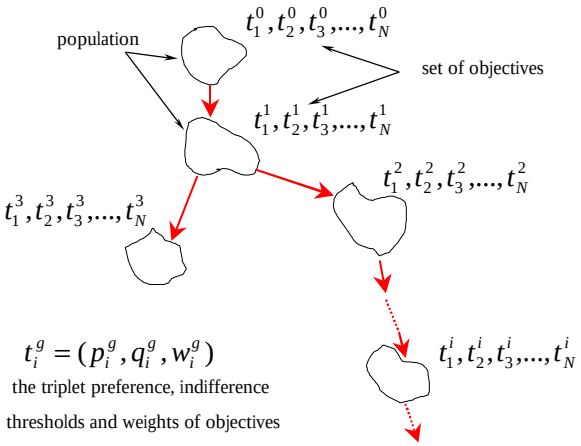

Figure 5.6. Branching on population.

[3] GAs have the feature that if repeatedly run on the same problem it will get a range of different results with varying quality. Even if this variability is often less than with classical methods, it still may be significant. In general, this variability can be reduced by running the GA a few times on the same problem, and taking the best result.
In practice the GA is run several times from beginning to end. Suppose, however, that the GA is ran in stages, that is, stopping after every n iterations and saving the current population. This process can be seen as an operator that generates one population from another. Because of the stochastic nature of this operator, it can be applied to the same 'parent' population (that is, go back and restart from the same saved population) many times, and it will generate many different 'child' populations. This operator can also be applied to one or more child populations. In this way we can generate a tree of populations.

Each criterion is attributed a triplet (w, p, q) where:

p is the preference threshold
If the absolute value of difference between two solutions is higher than *p*, that means that this difference is significant, and the solution representing the highest performance is better (preferred) than the other.
q is the indifference threshold
If the absolute value of difference between two solutions is lower than *q*, that means that this difference is not significant, and the two solutions are practically equivalent.

w is the weight
The weight *w* or in other words *coefficient of importance* means that if a criterion is attributed a weight of 3, and another a weight of 2, that means that 2 (respectively 3) points gained with the first criterion can be compensated by 4 (respectively 5.5) points gained in the second. Is supposed that the values *p* and *q* are the same for the two criterion.

Let:

- $t_i^g = (p_i^g, q_i^g, w_i^g)$ the triplet *p, q* and *w* of objective *i* at the generation *g,*
- $T^j = (t_1^j, t_2^j, t_3^j, ..., t_N^j)$ a set of triplets at generation *j,*
- *N* is the number of objectives.

The procedure starts by assigning a triplet to each criterion (T^i a set of triplets at generation *i*), GA is then run for a certain number of iterations g_i. The population obtained after g_i iterations (generations) is then analysed by the user (quality of the objectives, the global quality of the best solution, ...). If the user is unsatisfied by the solution, the triplets assigned to the different criterion can be modified. The GA is re-run again using the new values of the triplets (T^{i+1}). This technique of branching on populations helps to guide the GA to deal with MOPs looking for the compromise between objectives.

A *set of triplets* at generation *j* is obtained by applying the function M_i^j on the set of triplets at generation *i*:

$$t_1^j, t_2^j, t_3^j, ..., t_N^j = M_i^j(t_1^i, t_2^i, t_3^i, ..., t_N^i)$$

For instance suppose we have two objectives and let the two triplets at generation 0: $t_1^0 = (1, 0, 1), t_2^0 = (1, 0, 0)$ and let the modifying function M_0^1 which transforms the triplets of generation 0 to those of generation 1: $t_1^1 = (1.0, 0.5), t_2^1 = (1.0, 0.5)$.

The population at generation *j* is obtained running the GA and using its corresponding set of triplets $t_1^j, t_2^j, t_3^j, ..., t_N^j$ as illustrated in Figure 5.7.

$$P^j = M_i^j(P^i)$$

where :

- M_i^j is the function modifying triplets of generation *i* to triplets of generation *j*,
- P^i is the population at generation *i*,
- $M_i^j(P^i)$ transforms population of generation *i* using triplets $t_1^j, t_2^j, t_3^j, ..., t_N^j$.

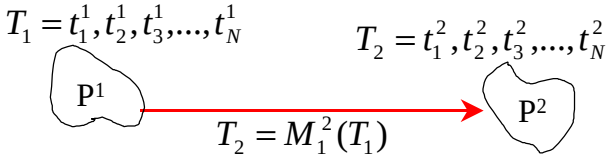

Figure 5.7. Evolving from generation 1 to generation 2.

The term *generation* is referred to the *order* of the modification. The modification function is set by the DM after analysing the quality of the best solution for the given set of triplets. The whole process of optimisation is composed by a set of *R* runs of the GA using different set of triplets. If the results of a modification cannot be accepted, due to an inappropriate setting of the triplets, the DM has the choice between:

- start the run from the beginning,
- go back to the last triplets and restart from the last population.

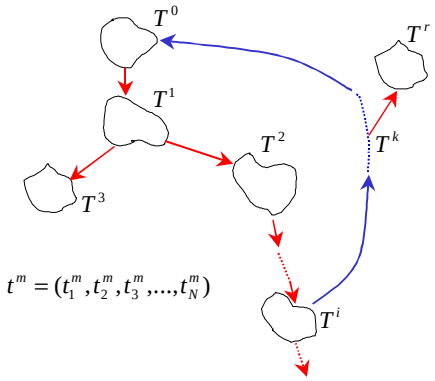

Figure 5.8. Cycling approach.

The modification function must be *reversible*. The *reversibility* means that a given set of triplets yield a same population (regardless of the stochastic behaviour of GAs). Abstraction is done from which population we start. In other words results obtained using a given set of triplets is independent of the starting population. Figure 5.8 illustrates our words. The population obtained using the triplets T^0 must be more or

less the same as the population obtained using the sequence of triplets T^0 T^1 T^2 ... T^i... T^k... T^0. The set of triplets allows to guide the GA during the search phase.

The 'branching on population' allows the DM to explore easily the search space. In addition to the GA's which permits to generate a *set* of populations, the method search in the *space of populations* dealing with the DM's *preferences*. In the same time the method can be fruitful to test the *robustness* of search methods.

The branching factor (the number of changes of a given number of triplets) in the case of k objectives is $N = 2^{3*k} - 1$ (*64* in the case of *two* objectives). It is the number of different groups composed by n objects (the factor *3* is due to the fact that for each objective we have a triplet p, q, w). These changes are not always possible and are far to be realistic. The backtracking technique can be used to go back to an old population. The step of variation (number of generations of each set of triplets) as well as the variance of variables depends on the size of problem at hand. In the next section an academic example is introduced to bring our words into play.

6. An academic example

Here, an *academic* MOP is addressed using the proposed method. It is a grouping problem with two totally *conflicting* objectives. The first objective is maximal if the second objective is minimal and vice-versa. A set of N objects have to be grouped in a set of groups. Let $Objs_i=\{i / i \hat{I} [0..N]\}$ be the set of objects in group i. The size of each object is equal to its identity.

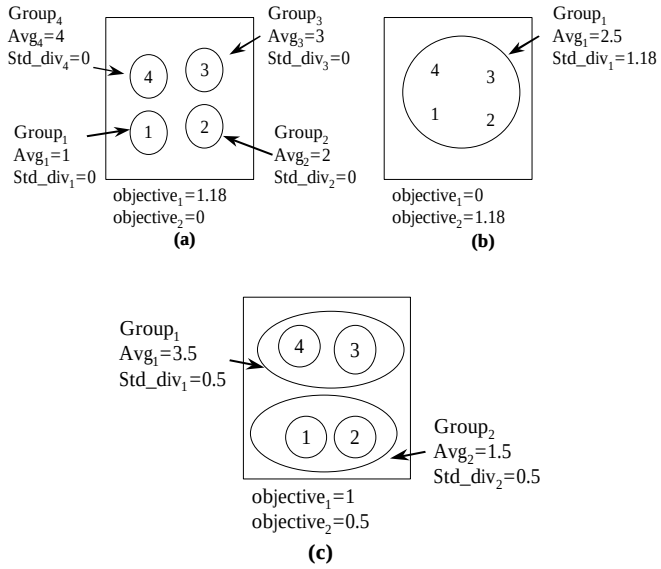

Figure 5.9. One object by group (a) and one group for all objects (b) and two objects by group (c) solutions.

A graphical representation of the problem is given in Figure 5.9. In the first solution each object {1, 2, 3, 4} is put in its own group while in the second solution all objects are grouped in the same group. Each group is characterised by its average and its standard deviation. The average of a given group is the sum of the identity of its objects divided by the number of its items (size). For instance in Figure 5.9 (a) the average of the group 4 is 4 since it is composed only by the object 4 while its standard deviation is null.

The aim is to optimise the two following objectives:

- minimise the standard deviation on the average of groups,
- minimise the average of the standard deviations.

Let's $NbGrp$ the number of groups of a given solution, and $size_i$ the size of each group.

The average of a group i is the sum of the size of objects of the group divided by the its size. It is given by:

$$Avg_i = \frac{\sum_j^{size_i} Objs_i[j]}{size_i}$$

The average of a given solution is the sum of the average of groups divided by the number of groups and it is given by:

$$AVG = \frac{\sum_{i=0}^{NbGrp} Avg_i}{NbGrp}$$

The standard deviation of a group indicates how closely the size of objects are clustered around the average:

$$Std\_div_i = Sqrt(\frac{\sum_{i=0}^{size_i} (Objs_i[j] - Avg_i)^2}{size_i})$$

The first objective is the standard deviation of the standard deviation of the groups:

$$Minimize: objective1 = Sqrt(\frac{\sum_{i=0}^{NbGrp} (Avg_i - AVG)^2}{NbGrp})$$

The second objective is the average of the different standard deviation:

$$Minimize: objective2 = \frac{\sum_{i=0}^{NbGrp} Std\_div_i}{NbGrp}$$

The formulation of the two objectives show that (see Figure 5.9):

- the first objective takes the value '0' if there is only one group and in the same time the second objective takes its maximal value.
- the second objective takes the value '0' if there only one object by group while the first objective takes its maximal value[4].

Figure 5.10 presents the evolution of the two objectives where the second objective is neglected (w1=1.0, w2=0.0). The word 'neglected' mean that the weight or the preference attributed to the given objective is set to zero. This figure shows that the method tends to minimise the objective 1, and ignores the other.

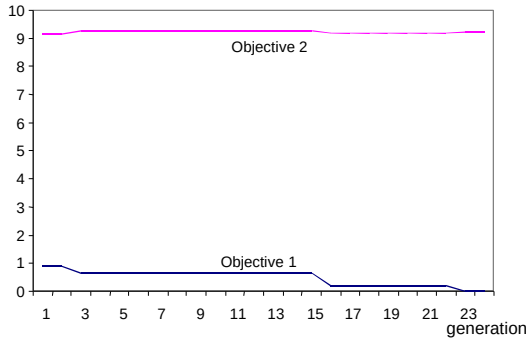

Figure 5.10. Evolution of objective 1 in case the objective 2 is neglected.

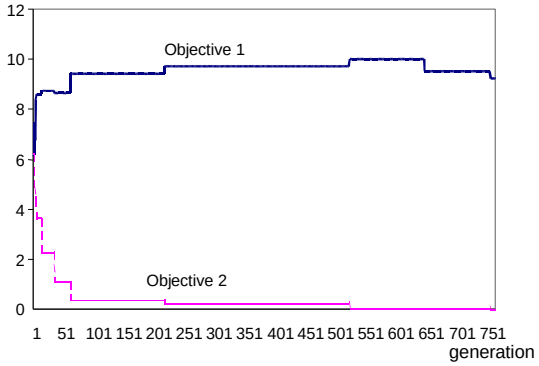

Figure 5.11. Evolution of objective 2 in case the objective 1 is neglected.

[4] Note, that a solution composed by two groups where the first group contains the objects {1, 4} and the second {2, 3} takes the value '0' for the first objective and the value '1' for the second objective. This solution is not optimal in case the aim is to find the minimum value of objective 1 and the maximal value of the objective 2.

Figure 5.11 represents the evolution of the two objectives where the objective 1 is neglected. The weight attributed to objective 1 is null (w1=0.0, w2=1.0). The figure shows that the method tends to minimise the objective 2, and pays less attention to the evolution of the objective 1. In fact, setting the weight of a given objective to zero indicates to the MO-GGA to give it less importance. Thus, minimising the objective 2 tends to maximise the objective 1 and vice-versa.

In case one wants to optimise the two objectives, a simple way is to set the weights to (w1=0.5, w2=0.5). Figure 5.12 shows that the two objectives are conflicting, that is, minimising one objective tends to maximise the other.

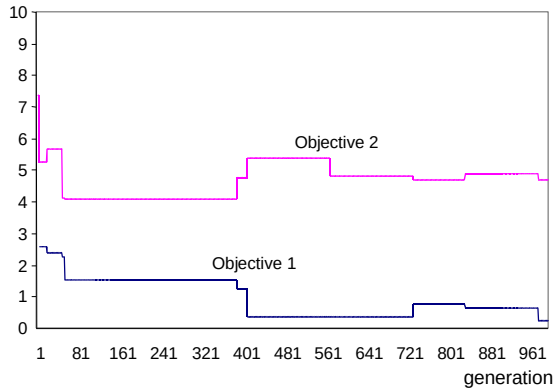

Figure 5.12. Evolution of two objectives having the same preference.

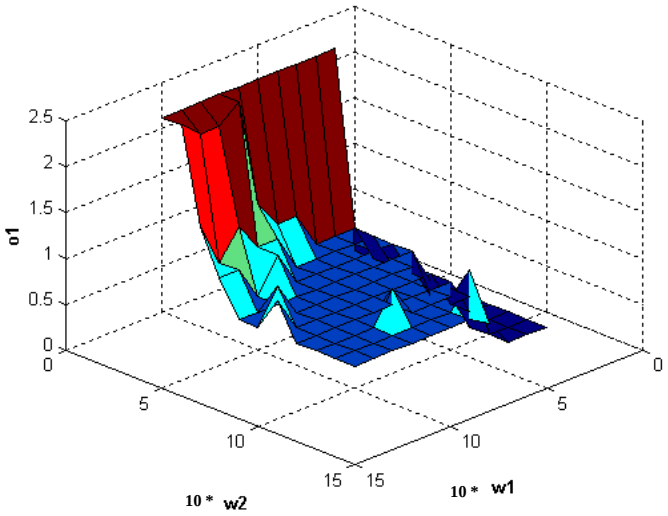

Figure 5.13. Evolution of objective 1 with preferences of the two objectives.

The evolution of objective 1 (see Figure 5.13) and objective 2 (Figure 5.14) for different values of w1 and w2, let us conclude that the weights given to the different objectives allow to guide the algorithm. Figure 5.15 represents the evolution of the two objectives for different values of w1 and w2. It shows that for small values of

w1, the algorithm tends to optimise objective 2 and for small values of w2, the algorithm tends to optimise objective 1. For similar values of w1 and w2, the algorithm optimises the two objectives at the same time. Thus, seeking to optimise an objective consists in choosing good preferences.

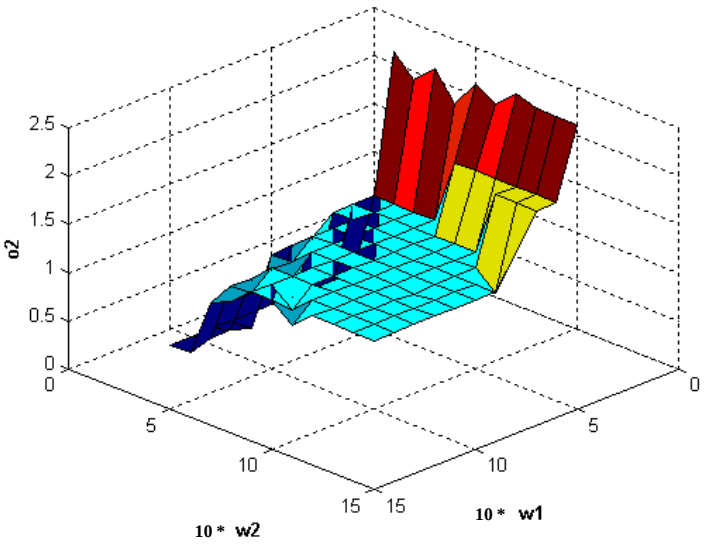

Figure 5.14. Evolution of objective 2 with preferences of the two objectives.

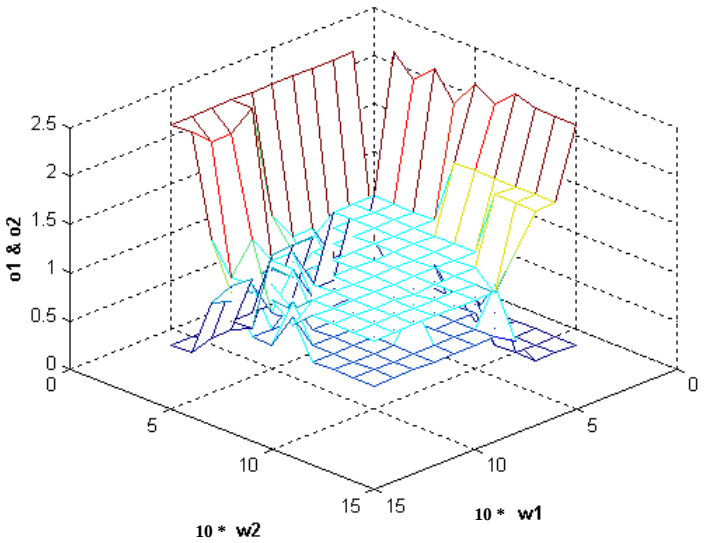

Figure 5.15. Evolution of objective 1 and 2 with their preferences.

In order to verify the idea of branching on population the following test was done. For the same problem, we started with the given preferences (w1=1.0, w2=0.0). The algorithm was stopped after *180* generation, then the preferences were set to

(w1=0.0, w2=1.0). Again, the algorithm was stopped at generation *570*, the preferences were set to (w1=1.0, w2=0.0) (see Figure 5.16 and Table 2). The graphic shows that the best solution of the population as well as the search direction changes once the preferences given to the objectives change. The fact that the population switch quickly from one direction to another is attributed to the *mutation* which makes a kind of *diversity*[5] in the population.

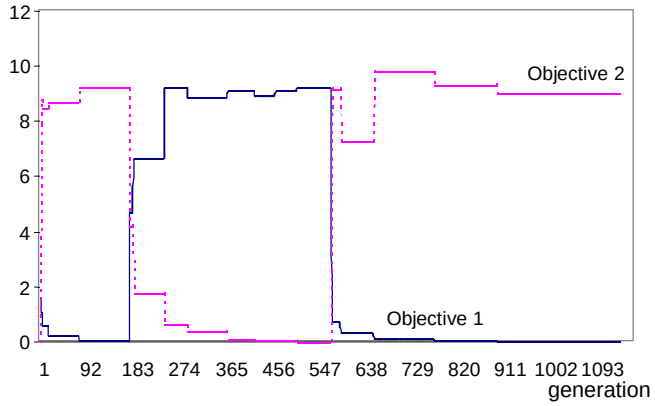

Figure 5.16. Changing the search direction by changing preferences.

generation	objective 1	objective 2		generation	objective 1	objective 2
2	1.59	7.92		291	8.87	0.38
3	1.59	7.92		368	9.1	0.11
4	1.02	8.78		420	8.89	0.06
5	1.02	8.78		462	9.11	0.03
6	0.56	8.51		504	9.23	0
18	0.22	8.71		570	5.4	7.27
78	0.06	9.21		572	0.75	9.19
177	4.67	4.19		589	0.32	7.3
183	4.67	4.19		655	0.12	9.81
184	6.67	1.75		772	0.07	9.35
245	9.21	0.62		896	0	9.02
290	9.21	0.62				

Table 2. Critical values of the two objectives.

These results show that the proposed method respects the user's preferences regarding the optimisation objective.

[5] The process of diversity loss is often the cause of premature convergence which is the early convergence on an inferior local maximum. A large number of existing techniques are used to maintain diversity in GAs. These include maintaining large population sizes, employing low reproductive or parent-selection pressures, applying mutation, restarting the GA, employing parallel populations, and niche-formation techniques.

7. Summary and applications

We *cannot* say 'anybody who is optimising is engaged in multiple objective optimisation, since it can be either single objective or multiple objective, and single objective functions can in some cases replace multiple objective functions'. Thus, the proposed GA approach to MOPs is also valid approach for problems having non proportional conflicting objectives, otherwise the weighed sum approach can do good job.

The idea of Pareto optimisation is to provide the DM with a representative set of solutions from the Pareto optimal front. The DM can see the tradeoffs that have to be made in choosing a solution, rather than asking these to be fixed through assignment of weights beforehand. The problem is the number of solutions the DM has to choose among. The human cannot easily decide among more than a few solutions, and the Pareto frontier most of the time is composed by many non-dominated solutions. Thus, we settled for proposed method rather than for the Pareto optimality.

We introduced a new paradigm to deal with multiple objective using evolutionary computation methods. The method is based on a the MO-GA, the genetic algorithm combined with the MCDA method. The MCDA method is based on the PROMETHEE II method. The method was chosen due to its simplicity, other MCDA methods can be used as well. We showed how the method can deal with the preferences, simply by adjusting the weight of the different objectives.

The proposed method will be used to two multiple objective grouping problems, namely the assembly line balancing (ALB) for manual assembly lines and the resource planning (RP) for hybrid assembly lines.

8. References

(Bentley, 1999) Bentley P. J., (ed.) 'Evolutionary design by computers', Academic Press Ltd., London, 1999.

(Brans, 1994) Brans J.-P. and Mareschal B., 'The PROMCALC & GAIA decision support system for multi-criteria decision aid', Decision Support Systems, North-Holland, Vol. 12, pp. 297-310, 1994.

(Coello, 1999) Coello C.A.C., 'A comprehensive survey of evolutionary-based multiobjective optimisation', Knowledge and Information Systems, Vol. 1(3), pp. 129-156, 1999.

(Cvetkovic, 1999) Cvetkovic D. and Parmee I.C., 'Use of preferences for GA-based multi-objective optimisation', Proceedings of GECCO'99, Orlando USA, pp. 1504-1509, 1999.

(Deb, 1999) Deb K., 'Multi-objective genetic algorithms: problem difficulties and construction of test problems', Evolutionary Computation, Vol. 7(3), pp. 205-230, 1999.

(Falkenauer, 1998) Falkenauer E., "Genetic algorithms and grouping problems", John Wiley & Sons Inc., Chichster, First Edition, 1998.

(Fonseca, 1998) Fonseca C. M. and Fleming P.J., "Multiobjective optimisation and multiple constraint handling with evolutionary algorithms: A unified formulation", IEEE Transactions on Systems, Man and Cybernetics, Vol. 28(1), pp. 26-47, 1998.

(Fonseca, 1995) Fonseca C.M. and Fleming P.J., "An overview of evolutionary algorithms in multiobjective optimisation", Evolutionary Computation, Vol. 3(1), pp. 1-16, 1995.

(Fourman, 1985) Fourman M. P., "Compaction of symbolic layout using genetic algorithms", in Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms, Lawrence Erlbaum, pp. 141-153, 1985.

(Goldberg, 1989) Goldberg D. E., "Genetic algorithms in search, optimisation and machine learning", Addison-Wessley, 1989.

(Hajela, 1992) Hajela, P. and Lin, C.Y., "Genetic search strategies in multicriterion optimal design", Structural Optimisation, Vol. 4, pp. 99-107, 1992.

(Holland, 1975) Holland J.H., "Adaptation in natural and artificial systems", University of Michigan Press, Ann Arbor, 1975.

(Horn, 1993) Horn J. and Nafpliotis N., "Multiobjective optimisation using the niched pareto genetic algorithm", Technical Report, IlliGAL Report 93005, University of Illinois, Urbana, Illinois, USA, 1993.

(Jaszkiewicz, 1998) Jaszkiewicz A., "Genetic local search for multiple objective combinatorial optimization", Research report, Institute of Computing Science, Poznan University of Technology, RA-014/98, 1998.

(Kursawe, 1991) Kursawe F., "A variant of evolution strategies for vector optimisation", in Parallel Problem Solving from Nature. 1st Workshop, PPSN I (H. P. Schwefel and R. Männer, eds.), Lecture Notes in Computer Science, Berlin Germany, Springer-Verlag, Vol. 496, pp. 193-197, 1991.

(Murata, 1999) Murata T., Ishibuchi H. and Gen M., "Specification of local search directions in genetic local search algorithms for multi-objective optimisation problems", Proceedings of GECCO'99, Orlando USA, pp. 441-448, 1999.

(Pahl, 1996) Pahl G. and Beitz W., "Engineering design–a systematic approach", Springer-Verlag, Berlin, 1996.

(Pareto, 1988) Pareto V. "Cours d'economie politique", F. Rouge, Lausanne, Vol. I and II, 1988.

(Ritzel, 1994) Ritzel B.J., Eheart J.W. and Ranjithan S., "Using genetic algorithms to solve a multiple objective groundwater pollution containment problem", Water Resources Research, Vol. 30, pp. 1589-1603, 1994.

(Rosenberg, 1967) Rosenberg R.S., "Simulation of genetic populations with biochemical properties", Ph.D. Thesis, University of Michigan, Ann Harbor, Michigan, 1967.

(Roy, 1968) Roy B., "Classement et choix en présence de points de vue multiples (la méthode Electre)", Revue Française d'Informatique et de Recherche Opérationnelle, Vol. 8, pp. 57-75, 1968.

(Schaffer, 1985) Schaffer J.D., "Multiple objective optimisation with vector evaluated genetic algorithms", In Genetic Algorithms and their Applications: Proceedings of the First ICGA, Lawrence Erlbaum, pp. 93-100, 1985.

(Schwefel, 1981) Schwefel H.P., "Numerical optimisation of computer models", Great Britain: John Wiley and sons, 1981.

(Srinivas, 1994) Srinivas N. and Deb K., "Multiobjective optimisation using nondominated sorting in genetic algorithms", Evolutionary Computation, Vol. 2, pp. 221-248, 1994.

(Steinberg, 1999) Steinberg L. and Rasheed K., "Optimizing by searching a tree of populations", Proceedings of GECCO 99, Orlando USA, pp. 1723-1730, 1999.

(To, 1997) To T. B. and Korn U., "MOBES: a multiobjective evolution strategy for constrained optimisation problems", The Third International Conference on Genetic Algorithms Mendel 97, Brno Czech Republic, pp. 176-182, 1997.

(Van Veldhuizen, 1999) Van Veldhuizen D.A., "Multiobjective evolutionary algorithms: classifications, analyses, and new Innovations", Ph.D. Thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1999.

(Wilson, 1993) Wilson P. B. and Macleod M. D., "Low implementation cost IIR digital filter design using genetic algorithms", IEE/IEEE Workshop on Natural Algorithms in Signal Processing, Chelmsford, U.K., Vol. 4, pp. 1-8, 1993.

(Ziztler, 1999) Ziztler E., "evolutionary algorithms for multiobjective optimization: methods and applications", Ph.D. Thesis, Swiss Federal Institute of Technology (ETH), Zurich Switzerland, 1999.