CHAPTER III

MULTI-OBJECTIVE OPTIMIZATION PROBLEM

3.1 Multi-objective optimization problem

3.1.1 Basic Concepts and Terminology

When an optimization problem involves more than one objective function, the task of finding one or more optimum solutions is known as *multi-objective optimization*. In the parlance of management, such search and optimization problems are known as multiple criterion decision-making (MCDM). Since multi-objective optimization involves multiple objectives, it is intuitive to realize that single-objective optimization is degenerate case of multi-objective optimization. However, there is another reason why more attention is now being focused on multi-objective optimization – a matter we will discuss in the next paragraph.

Most real-world search and optimization problems naturally involve multiple objectives. The extremist principle mentioned above cannot be applied to only one objective, when the rest of the objectives are also important. Different solution may produce trade-offs (conflicting scenarios) among different objective.

Def. 1: (Multi-objective Optimization Problem) A general MOP includes a set of n parameters (decision variables), a set of k objective functions, and a set of m constraints. Objective functions and constraints are functions of the decision variables. The optimization goal is to

$$\begin{aligned} &\textit{Maximize} && y = f(x) = f_1(x), f_2(x), \ldots\ldots f_k(x) \\ &\textit{Subject to} && e(x) = e_1(x), e_2(x), \ldots\ldots e_k(x) \leq 0 \\ &\textit{Where} && x = (x_1, x_2, \ldots\ldots, x_n) \in X \\ & && y = (y_1, y_2, \ldots\ldots, y_k) \in Y \end{aligned} \qquad (3.1)$$

Where x is the decision vector, y is the objective vector, X is denoted as the decision space, and Y is called the objective space. The constraints e(x) $\leq$0 determines the set of feasible solutions.

Def. 2: (Feasible Set) the feasible set X f is defined as the set of decision vectors x that satisfy the constraints e(x) :

$$X_f = \{x \in X \mid e(x) \leq 0\} \tag{3.2}$$

The image of X_f , i.e., the feasible region in the objective space, is denoted as without loss of generality, a maximization problem is assumed here. For minimization or mixed maximization/minimization problems the definitions presented in this section are similar. Consider again the above example and assume that the two objectives performance (f_1) and cheapness (f_2), the inverse of cost, are to be maximized under size constraints (e_1). Then an optimal design might be an architecture which achieves maximum performance at minimal cost and does not violate the size limitations. If such a solution exists, we actually only have to solve a single-objective optimization problem (SOP). The optimal solution for either objective is also the optimum for the other objective. However, what makes MOPs difficult is the common situation when the individual optima corresponding to the distinct objective functions are sufficiently different. Then, the objectives are conflicting and cannot be optimized simultaneously. Instead, a satisfactory trade-off has to be found. In our example, performance and cheapness are generally competing: high-performance architectures substantially increase cost, while cheap architectures usually provide low performance. Depending on the market requirements, an intermediate solution (medium performance, medium cost) might be an appropriate trade-off. This discussion makes clear that a new notion of optimality is required for MOPs.

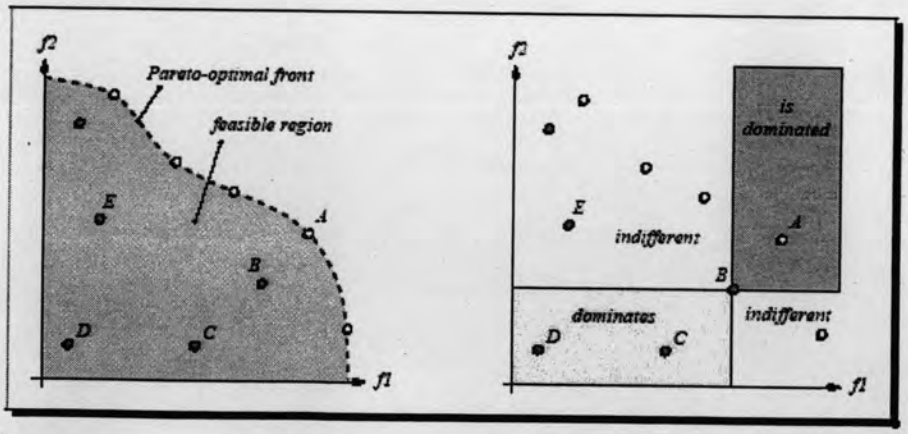

Figure 3.1: Illustrative example of Pareto optimality in objective space (left) and the possible relations of solutions in objective space (right).

In single-objective optimization, the feasible set is completely (totally) ordered according to the objective function *f* for two solutions *a*, *b* ∈ X_f either *f (a)* ≤ *f (b)* or *f (b)* ≤ *f (a)*. The goal is to find the solution (or a solution) that gives the maximum value of *f*. However, when several objectives are involved, the situation changes: is X_f, in general, not totally ordered, but partially ordered. This is illustrated in Fig. 3.1 on the left. The solution represented by point *B* is better than the solution represented by point *C*: it provides higher performance at lower cost. It would be even preferable if it would only improve one objective, as is the case for *C* and *D*: despite equal cost, *C* achieves better performance than *D*. In order to express this situation mathematically, the relations =, ≥ and > are extended to objective vectors by analogy to the single-objective case.

Def. 3: For any two objective vectors u and v,

$$\begin{aligned} u = v &\text{ if } \forall_i \in \{1, 2, \ldots, k\} : u_i = v_i \\ u \geq v &\text{ if } \forall_i \in \{1, 2, \ldots, k\} : u_i \geq v_i \\ u > v &\text{ if } u \geq v \wedge u \neq v \end{aligned} \qquad (3.3)$$

The relations ≤ and < are defined similarly.

Using this notion, it holds that *B* > *C*, *C* > *D*, and, as a consequence, *B* > *D*. However, when comparing *B* and *E*, neither can be said to be superior, since $B \leq E$ *and* $E \leq B$. Although the solution associated with *E* is cheaper, it provides lower performance than the solution represented by *B*. Therefore, two decision vectors ***a, b*** can have *three* possibilities with MOPs regarding the ≥ relation (in contrast to two with SOPs): *f (a)* ≥ *f(b)*, *f (b)* ≥ *f (a)*, or *f (a)* < *f (b)* ∧ *f (b)* < *f (a)*. Here, the following symbols and terms are used in order to classify the different situations.

Def. 4: (Pareto Dominance) for any two decision vectors a and b,

$$\begin{aligned} &\mathbf{a \succ b \text{ (a dominates b) if } f(a) > f(b)} \\ &\mathbf{a \rhd b \text{ (a weakly dominates b) if } f(a) \geq f(b)} \\ &\mathbf{a \approx b \text{ (a is indifferent to b) if } f(a) < f(b) \wedge f(b) < f(a)} \end{aligned} \qquad (3.4)$$

The definitions for a minimization problem ($\prec, \lhd, \approx$) are analogical.

In Fig 3.1 on the right, the light gray rectangle encapsulates the region in objective space that is dominated by the decision vector represented by *B*. The dark gray rectangle contains the objective vectors whose corresponding decision vectors dominate the solution associated with *B*. All solutions for which the resulting objective vector is in neither rectangle are indifferent to the solution represented by *B*.

Based on the concept of Pareto Dominance, the optimality criterion for Multi-objective optimization problems can be introduced. Still referring to Fig. 3.1, *A* is unique among *B*,*C*, *D*, and *E*: its corresponding decision vector *a* is not dominated by any other decision vector. That means, *A* is optimal in the sense that it cannot by improved in any objective without causing degradation in at least one other objective. Such solutions are denoted as *Pareto optimal*; sometimes also the term *non-inferior* is used.

Def. 5: (Pareto Optimality) A decision vector x $\in X_f$ is said to be non-dominated regarding a set A $\subseteq X_f$ if

$$\exists a \in A : a \succ x \qquad (3.5)$$

If it is clear within the context which set A is meant, it is simply left out. Moreover, x is said to be Pareto optimal if xx is non-dominated regarding X_f .

In Fig. 3.1 the white points represent Pareto-optimal solutions. They are indifferent to each other. This makes the main difference to SOPs clear: there is no single optimal solution but rather a set of optimal trade-offs. None of these can be identified as better than the others unless preference information is included (e.g., a ranking of the objectives).

The entirety of all Pareto-optimal solutions is called the *Pareto-optimal set*; the corresponding objective vectors form the *Pareto-optimal front* or *surface*.

Def. 6: (Non-dominated Sets and Fronts) Let A$\subseteq X_f$. The function p(A) gives the set of non-dominated decision vectors in A:

$$p(A) = \{a \in A \mid a \text{ is non-dominated regarding A}\} \qquad (3.6)$$

The set *p(A)* is the non-dominated set regarding A, the corresponding set of objective vectors f (p(A)) is the non-dominated front regarding A. Furthermore, the set X_p = p(X_f) is called the Pareto-optimal set and the set Y_p = f (X_p) is denoted as the Pareto-optimal front.

The Pareto-optimal set comprises the globally optimal solutions. However, as with SOPs there may also be local optima which constitute a non-dominated set within a certain neighborhood. This corresponds to the concepts of global and local Pareto-optimal sets introduced by Deb (1998, 1999a):

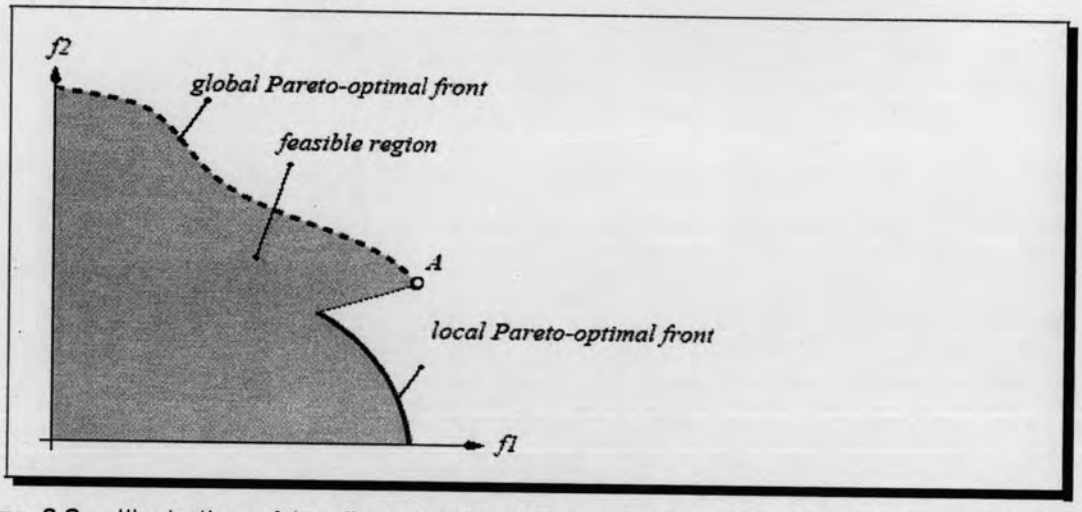

Figure 3.2: Illustration of locally optimal solution sets and globally optimal solution sets in objective space.

Def. 7: Consider a set of decision vectors A $\subseteq X_f$.

1. The set A is denoted as a local Pareto-optimal set if

$$a \in A : \exists x \in X_f : x \succ a \wedge ||x \quad a|| < \varepsilon \wedge ||f(x) \quad f(a)|| < \delta \qquad (3.7)$$

Where $\|\cdot\|$ *is a corresponding distance metric and* $\varepsilon > 0$, $\delta > 0$.

2. *The set A is called a* global Pareto-optimal set *if*

$$a \in A : \exists x \in X_f : x \succ a \qquad (3.8)$$

The difference between local and global optima is visualized in Fig. 3.2. The dashed line constitutes a global Pareto-optimal front, while the solid line depicts a local Pareto-optimal front. The decision vectors associated with the latter are locally non-dominated though not Pareto-optimal, because the solution related to point *A* dominates any of them. Finally, note that a global Pareto-optimal set does not necessarily contain all Pareto-optimal solutions and that every global Pareto-optimal set is also a local Pareto-optimal set.

3.1.2 Search and Decision Making

In solving an MOP, two conceptually distinct types of problem difficulty can be identified (Horn 1997): search and decision making. The first aspect refers to the optimization process in which the feasible set is sampled for Pareto-optimal solutions. As with single-objective optimization, large and complex search spaces can make search difficult and preclude the use of exact optimization methods like linear programming (Steuer 1986). The second aspect addresses the problem of selecting a suitable compromise solution from the Pareto-optimal set. A human decision maker (DM) is necessary to make the often difficult trade-offs between conflicting objectives. Depending on how optimization and the decision process are combined, multi-objective optimization methods can be broadly classified into three categories (Hwang and Masud 1979; Horn 1997):

Decision making before search: The objectives of the MOP are aggregated into a single objective which implicitly includes preference information given by the DM.

Search before decision making: Optimization is performed without any preference information given. The result of the search process is a set of (ideally Pareto-optimal) candidate solutions from which the final choice is made by the DM.

Decision making during search: The DM can articulate preferences during the interactive optimization process. After each optimization step, a number of alternative trade-offs is presented on the basis of which the DM specifies further preference information, respectively guides the search.

The aggregation of multiple objectives into one optimization criterion has the advantage that the classical single-objective optimization strategies can be applied without further modifications. However, it requires profound domain knowledge which is usually not available. For example, in computer engineering design space exploration specifically aims at gaining deeper knowledge about the problem and the alternative solutions. Performing the search before decision making overcomes this drawback, but excludes preference articulation by the DM which might reduce the search space complexity. Another problem with this and also the third algorithm category might be the visualization and the presentation of non-dominated sets for higher dimensional MOPs (Cohon, 1985). Finally, the integration of search and decision

making is a promising way to combine the other two approaches, uniting the advantages of both. In this thesis, the focus is on multi-objective optimization methods that are capable of

1. Sampling intractably large and highly complex search spaces, and
2. Generating the exact Pareto-optimal set or approximations of it.

This is the first step in the direction of decision making during search and forms the basis for further research in this area.

3.2 Traditional Approaches

Classical methods for generating the Pareto-optimal set aggregate the objectives into a single, parameterized objective function by analogy to decision making before search. However, the parameters of this function are not set by the DM, but systematically varied by the optimizer. Several optimization runs with different parameter settings are performed in order to achieve a set of solutions which approximates the Pareto-optimal set. Basically, this procedure is independent of the underlying optimization algorithm.

Some representatives of this class of techniques are the weighting method (Cohon , 1978), the constraint method (Cohon, 1978), goal programming (Steuer, 1986), and the minmax approach (Koski, 1984). In place of the various methods, the two first mentioned are briefly discussed here.

3.2.1 Weighting Method

The original MOP is converted to an SOP by forming a linear combination of the objectives:

$$\begin{aligned} &\textbf{Maximize} && y = f(x) = w_1.f_1(x) + w_2.f_2(x) + \ldots\ldots + w_k.f_k(x) \\ &\text{Subject to} && x \in X_f \end{aligned} \qquad (3.9)$$

The w_i are called weights and, without loss of generality, normalized such that $\sum w_i = 1$. Solving the above optimization problem for a certain number of different weight combinations yields a set of solutions.

On condition that an exact optimization algorithm is used and all weights are positive, this method will only generate Pareto-optimal solutions which can be easily shown. Assume that a feasible decision vector ***a*** maximizes *f* for a given weight combination and is not Pareto optimal. Then, there is a solution ***b*** which dominates *a*, i.e., without loss of generality $f_1(b) > f_1(a)$ and $f_i(b) \geq f_i(a)$ for $i = 2,\ldots, k$. Therefore, $f(b) > f(a)$, which is a contradiction to the assumption that $f(a)$ is maximum.

The main disadvantage of this technique is that it cannot generate all Pareto-optimal solutions with non-convex trade-off surfaces. This is illustrated in Fig.3.3

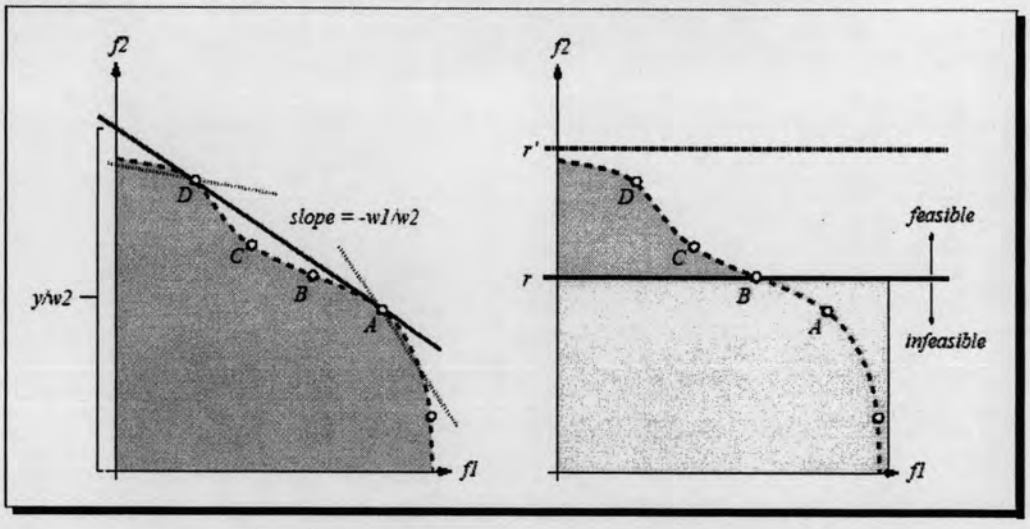

Figure 3.3: Graphical interpretation of the weighting method (left) and the constraint method (right).

This is illustrated in Fig. 3.3 based on the embedded system design example. For fixed weights w_1, w_2 solution *x* is sought to maximize $y = f(x) = w_1.f_1(x) + w_2.f_2(x)$. This equation can be reformulated as $f_2(x) = \frac{-w_1}{w_2} f_1(x) + \frac{y}{w_2}$, which defines a line with slope $\frac{-w_1}{w_2}$ and intercept $\frac{y}{w_2}$ in objective space (solid line in Fig. 3.3). Graphically, the optimization process corresponds to moving this line upwards until no feasible objective vector is above it and at least one feasible objective vector (here *A* and *D*) is on it. However, the points *B* and *C* will never maximize *f*. If the slope is increased, *D* achieves a greater value of *f* (upper dotted line); if the slope is decreased, *A* has a greater *f* value than *B* and *D* (lower dotted line).

3.2.2 Constraint Method

Another technique which is not biased towards convex portions of the Pareto-optimal front transforms k- 1 of the k objectives into constraints. The remaining objective, which can be chosen arbitrarily, is the objective function of the resulting SOP:

$$\begin{aligned} &\text{Maximize} && y = f(x) = f_h(x) \\ &\text{Subject to} && e_i(x) = f_i(x) \geq \varepsilon_i \quad (1 \leq i \leq k, i \neq h) \\ & && x \in X_f \end{aligned} \tag{3.10}$$

The lower bounds, ε_i, are the parameters that are varied by the optimizer in order to find multiple Pareto-optimal solutions. As depicted in Fig. 3.3 on the right, the constraint method is able to obtain solutions associated with non-convex parts of the trade-off curve. Setting h = 1 and $\varepsilon_2 = r$ (solid line) makes the solution represented by A infeasible regarding the extended constraint set, while the decision vector related to B maximizes f among the remaining solutions. Fig. 3.3 also shows a problem with this technique. If the lower bounds are not chosen appropriately ($\varepsilon_2 = r'$), the obtained new feasible set might be empty, i.e., there is no solution to the corresponding SOP. In order to avoid this situation, a suitable range of values for the ε_i has to been known beforehand.

3.2.3 Discussion of Classical Methods

What makes traditional approaches attractive and why they are popular may be attributed to the fact that well-studied algorithms for SOPs can be used. For large-scale problems, hardly any *real* multi-objective optimization techniques had previously been available (Horn 1997). By contrast, in single-objective optimization a wide range of heuristic methods have been known that are capable of dealing with this complexity, e.g., random search algorithms (Torn and Zilinskas, 1989), stochastic local search algorithms (Horst and Pardalos, 1995), simulated annealing (T¨orn and ˇ Zilinskas, 1989), tabu search (Glover, Taillard, and de Werra, 1993), etc. However, the preceding sections on weighting and constraint methods show that some difficulties may also accompany classical optimization strategies.

- Some techniques, e.g., the weighting method, may be sensitive to the shape of the Pareto-optimal front.
- Problem knowledge may be required which may not be available.

Deb (1999b) mentions further potential problems with these approaches, i.e., application areas where their use is restricted. Moreover, classical methods all have in common that they require several optimization runs to obtain an approximation of the Pareto-optimal set. As the runs are performed independently from each other, synergies can usually not be exploited which, in turn, may cause high computation overhead. However, this again depends on the application. Recently, evolutionary algorithms have become established as an alternative. To classical methods through which i) large search spaces can be handled and ii) multiple alternative trade-offs can be generated in a single optimization run. Furthermore, they can be implemented in a way such that both of the above difficulties are avoided.

3.3 Evolutionary Algorithms

The term evolutionary algorithm (EA) stands for a class of stochastic optimization methods that simulate the process of natural evolution. The origins of EAs can be traced back to the late 1950s, and since the 1970s several evolutionary methodologies have been proposed, mainly genetic algorithms, evolutionary programming, and evolution strategies (Back, Hammel, and Schwefel 1997).All of these approaches operate on a set of candidate solutions. Using strong simplifications, this set is subsequently modified by the two basic principles of evolution: selection and variation. Selection represents the competition for resources among living beings. Some are better than others and more likely to survive and to reproduce their genetic information. In evolutionary algorithms, natural selection is simulated by a stochastic selection process. Each solution is given a chance to reproduce a certain number of times, dependent on their quality. Thereby, quality is assessed by evaluating the individuals and assigning them scalar fitness values. The other principle, variation, imitates natural capability of creating"new" living beings by means of recombination and mutation.

Although the underlying principles are simple, these algorithms have proven themselves as a general, robust and powerful search mechanism. Back, Hammel, and Schwefel (1997) argue that the most significant advantage of using evolutionary search lies in the gain of flexibility and adaptability to the task at hand, in combination with robust performance (although this depends on the problem class) and global search characteristics." Moreover, EAs seem to be especially suited to multi-objective optimization because they are able to capture multiple

Pareto-optimal solutions in a single simulation run and may exploit similarities of solutions by recombination. Some researchers suggest that multi-objective search and optimization might be a problem area where EAs do better than other blind search strategies (Fonseca and Fleming 1995b; Valenzuela-Rend´on and Uresti-Charre, 1997). Although this statement must be qualified with regard to the "no free lunch" theorems for optimization (Wolpert and Macready, 1997), up to now there are few if any alternatives to EA-based multi-objective optimization (Horn, 1997). The numerous applications and the rapidly growing interest in the area of multi-objective evolutionary algorithms (MOEAs) take this fact into account.

3.4 Approaches to multi-objective optimization

Although the fundamental difference between these two optimizations lies in the cardinality in the optimal set, from a practical standpoint a user needs only one solution, no matter whether the associated optimization problem is a single-objective or multi-objective. In the case of multi-objective optimization problem, the user is now is a dilemma. Which of these optimal solutions must one choose? Thus, in a multi-objective optimization, ideally the effort must be made in finding the set of trade-off optimal solutions by considering all objectives to be important. After a set of trade-off optimal solutions are found, a user can then use higher-level qualitative considerations to make a choice. In view of these discussions, we therefore suggest the following principle for an *ideal multi-objective optimization procedure*.

Step 1 Find multiple trade-off optimal solution with a wide range of values for objectives.
Step 2 Chose one of the obtained using higher-level information.

Figure 3.4 shows schematically the principles in an ideal multi-objective optimization procedure. In Step 1(vertical downwards), multiple trade-off solutions are found. Thereafter, in Step2 (horizontally, towards the right), higher-level information is used to choose one of trade-off solutions. With this procedure in mind, it is easy to realize that single-objective optimization is a degenerate case of multi-objective optimization with only one global optimal solution. Step 1 will find only one solution, thereby not requiring us to proceed to Step 2. In the case single-objective optimization with multiple global optima, both steps are necessary to first find all or

many of the global optima and then to choose one from then to choose one from them by using the higher-level information about the problem

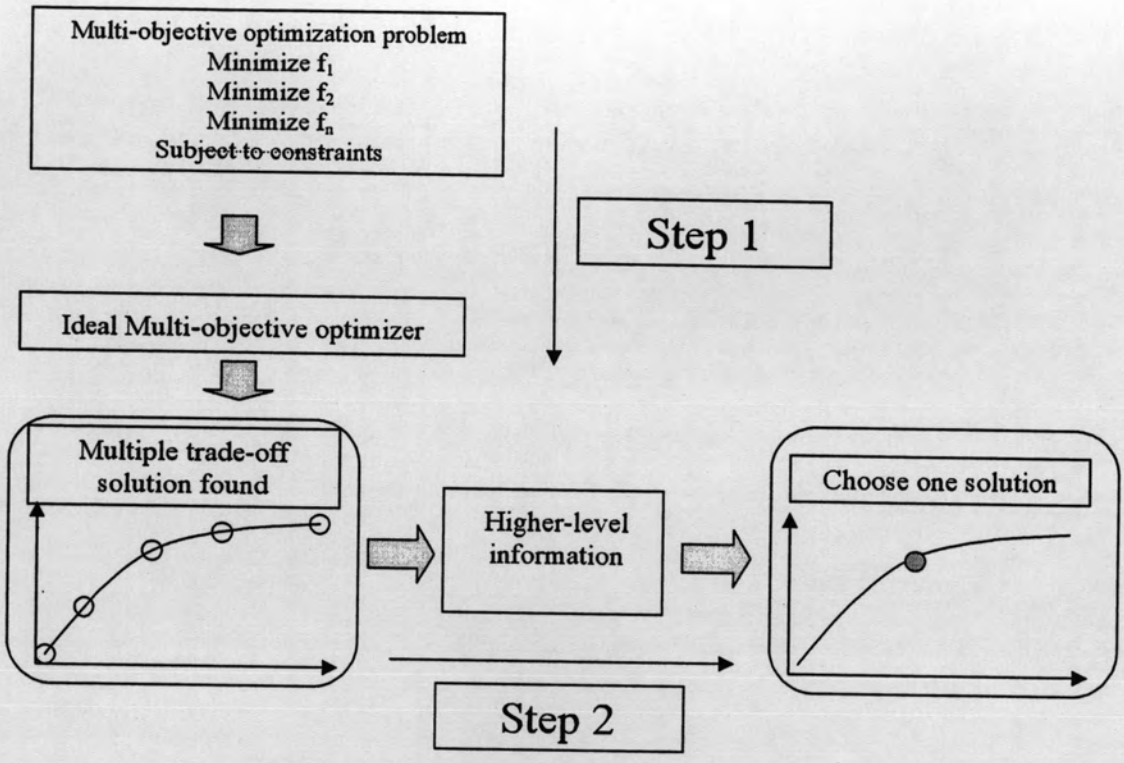

Figure 3.4: Schematic of an ideal multi-objective optimization procedure