1

Introduction to the Optimal Design of Complex Mechanical Systems

When a designer is able to simulate the physical behaviour of a system by means of a validated mathematical model, the subsequent task is that of defining the system model parameters (also called design variables, see Chap. 2) in order to obtain the desired system performances (also called objective functions, see Chap. 2).

Often such performances are conflicting, i.e. improving one implies the worsening of another, so a compromise has to be reached.

When the system model parameters are more than four or five and the system's objective functions are more than five or six, both the definition of parameters and the balancing of conflicting performances may become cumbersome or even impossible unless a special approach is adopted.

This Chapter will deal with the above-stated problem, i.e. the definition of system model parameters of complex systems, when conflicting performances have to be balanced. The design of complex systems will be accomplished by exploiting multi-objective programming (MOP), an optimisation theory pertaining to operational research (OR).

1.1 On the Optimal Design of Complex Systems

In order to introduce to the reader what is meant exactly by optimisation of a complex system, let us resort to an example. Let us imagine that a cantilever has to be optimally designed[1]. The example actually deals with a simple system, however, in this apparently simple problem, the main features pertaining to the optimisation of complex systems are present.

The cantilever, shown in Fig. 1.1, has a rectangular cross-section and a force acts at the free end. Let us assume that the optimisation problem to be solved is

[1]An effective *design process* should always produce an optimal solution, so *optimisation* and *design* are reputed to be the *same process* by the authors

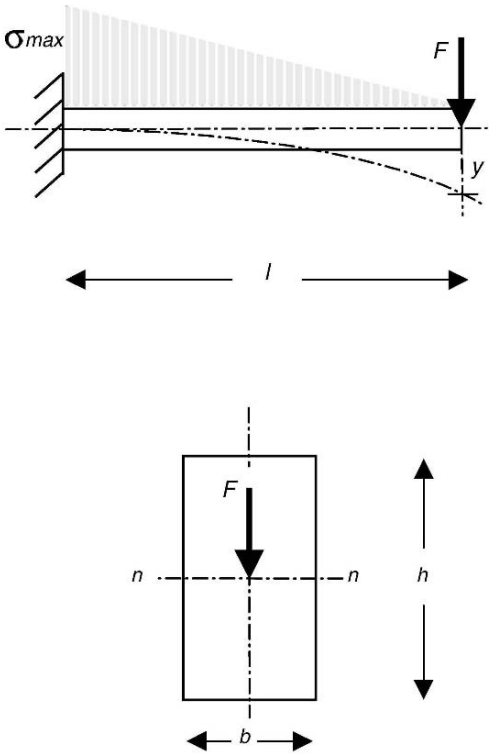

Fig. 1.1. Cantilever whose rectangular cross-section is to be defined in order to minimise both the cantilever mass and the cantilever deflection at the free end

- *finding the values of the design variables (length b and length h) defining the cantilever cross-section*[2]

in order to

- *minimise the cantilever mass*
- *minimise the cantilever deflection at its free end,*

subject to the following conditions

- *the maximum stress at the fixed end must be little than (or equal to) the admissible stress*
- *elastic stability must be guaranteed (i.e. buckling must be avoided).*

A designer should chose the values defining the cross-section of the cantilever in order to get it as light and stiff as possible, avoiding both a failure (due to too high stress) and elastic instability.

[2] b and h may vary, respectively, within two well-defined ranges

In mathematical form, the above optimisation problem may be stated as follows:

- *Given*
  - l the cantilever length (m)
  - b the beam cross-section width (m)
  - h the beam cross-section height (m)
  - J $= \frac{1}{12}bh^3$, the flexural moment of inertia of the section (n–n axis) (m^4)
  - F the force applied at the cantilever free end (see Fig. 1.1) (N)
  - σ_E the material yield stress (MPa)
  - η safety coefficient ($\geq$1)
  - $\sigma_{adm} = \sigma_E/\eta$ the admissible stress at the cantilever fixed end (MPa)
  - E the material modulus of elasticity (Young's modulus) (MPa)
  - G the material modulus of tangential elasticity (MPa)
  - ρ the material density (kg/m^3)
- *and defining*
  - $m = \rho bhl$ the cantilever mass (kg)
  - $y = \frac{1}{3}\frac{Fl^3}{EJ} = 4\frac{Fl^3}{Ebh^3}$ the deflection at the free end due to load F (m)
  - $\sigma_{max} = 6\frac{Fl}{bh^2}$ the maximum stress located at the top of the cross-section at the fixed end of the cantilever (MPa)
  - $F_{cr} = \frac{k_1 b^3 h}{l^2}\sqrt{\left(1 - k_2\frac{b}{h}\right)EG}$ the critical load (see [270]) (N)
- *find b and h such that*

$$b_{min} \leq b \leq b_{max}$$
$$h_{min} \leq h \leq h_{max}$$

- *and such that*

$$\min \begin{pmatrix} m(b,h) \\ y(b,h) \end{pmatrix} = \begin{pmatrix} \rho bhl \\ 4\frac{Fl^3}{Ebh^3} \end{pmatrix} \tag{1.1}$$

- *subject to*

$$\sigma_{max} = 6\frac{Fl}{bh^2} \leq \sigma_{adm} = \sigma_E/\eta \tag{1.2}$$

$$F < F_{cr} = \frac{k_1 b^3 h}{l^2}\sqrt{\left(1 - k_2\frac{b}{h}\right)EG} \tag{1.3}$$

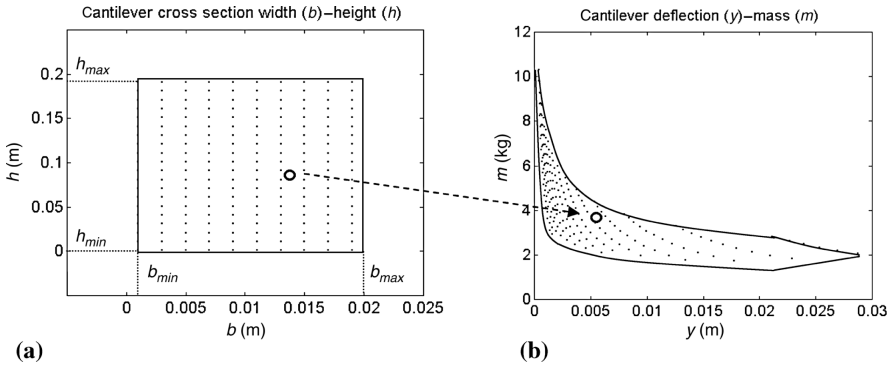

Fig. 1.2. (**a**) The cantilever cross-section width b and height h that are considered for the optimisation. (**b**) Cantilever mass $m(b,h)$ and cantilever deflection at the free end $y(b,h)$. The manifold in b accounts for the constraints on the maximum stress (1.2) and on the critical load (1.3). Data (symbols referring to (1.2) , (1.3), and Fig. 1.1): $b_{min} = 0.001\,\mathrm{m}$, $b_{max} = 0.020\,\mathrm{m}$, $h_{min} = 0.001\,\mathrm{m}$, $h_{max} = 0.200\,\mathrm{m}$, $\rho = 2,700\,\mathrm{kg/m^3}$ $E = 70,000\,\mathrm{MPa}$, $G = 27,000\,\mathrm{MPa}$, $\sigma_E = 160\,\mathrm{MPa}$, $\eta = 1$, $F = 1,000\,\mathrm{N}$, $l = 1\,\mathrm{m}$, $k_1 = 0.669$, $k_2 = 0.63$

The optimisation problem aims to minimise[3] the *objective functions* $(m(b,h),\ y(b,h))$ by selecting properly the *design variables*[4] (b,h) and by satisfying the constraints on F_{cr} and σ_{adm}.

In order to solve the problem, one may compute $m(b,h)$ and $y(b,h)$ as function of all possible combinations of b and h.

This is a naive but reasonable approach. In other words, if a couple of values b,h are selected, then, correspondingly, a couple of values $m(b,h)$, $y(b,h)$ can be computed. The values of b and h have to be varied, respectively, within the prescribed ranges $b_{min} \le b \le b_{max}$ and $h_{min} \le h \le h_{max}$. The inequalities (1.2) and (1.3) have to be evaluated for all possible combinations of b and h: if the inequalities are satisfied, the couple of values $m(b,h)$, $y(b,h)$ is kept, otherwise it is discarded.

In Fig. 1.2 the results of such a computation are shown. At one point in the rectangle on the plane b–h corresponds a point in the manifold on the plane y–m. The rectangular manifold on the plane b–h is transformed into the

[3] If, for a generic optimisation problem, the objective functions were to be *maximised*, changing their signs would transform the maximisation problem into a minimisation one, so the presented example dealing with minimisation is quite general

[4] b and h are precisely *parameters*, with the property that they have to be varied, thus they are *variable parameters*, which is a nonsense as parameters do have fixed values. To avoid misunderstanding these variable parameters are named *design variables* to distinguish them from actual parameters having fixed values

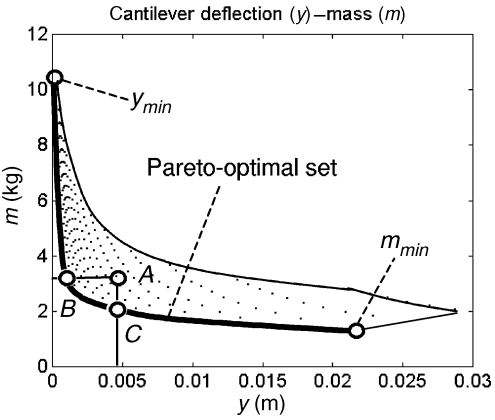

Fig. 1.3. Definition of the Pareto-optimal set in the objective functions domain. Data in Fig. 1.2

manifold on the plane y–m. In other words, a transformation is established between the domain of design variables and the domain of objective functions.

As all the possible combinations of design variables b, h have been used to generate $m(b,h)$ $y(b,h)$ (having verified that $m(b,h)$, $y(b,h)$ do satisfy the constraints (1.2) and (1.3)), the question is now how to find the values of b, h which minimise concurrently the mass m and the deflection y at the free end of the cantilever.

The definition of the solution of the addressed optimisation problem is not straightforward and requires a special reasoning.

Let us consider the couple of values $m_A(b_A, h_A)$, $y_A(b_A, h_A)$ referring to point A in Fig. 1.3. The couple of values b_A, h_A defines a cantilever cross-section which *should not* be considered by a designer, that is, the cantilever A is a wrong solution for the addressed optimisation problem.

The reason for this is readily explained. Let us consider point $B(m_B(b_B, h_B), y_B(b_B, h_B))$. It is $m_A(b_A, h_A) = m_B(b_B, h_B)$ but $y_B(b_B, h_B) < y_A(b_A, h_A)$: the two cantilevers have the same mass m but the deflection of the cantilever B is smaller than that of the cantilever A. Thus, the cantilever B is better than cantilever A. Similarly, for cantilever C, $m_C(b_C, h_C) < m_A(b_A, h_A)$ and $y_A(b_A, h_A) = y_C(b_C, h_C)$. Also the cantilever C is better than cantilever A because they have the same deflection but the mass of the cantilever C is smaller than that of the cantilever A. Point A is said to be *dominated* by B and C, i.e. B and C *dominate* A. By inspection of Fig. 1.3, A is also dominated by all the design solutions between B and C on the bold line, i.e. the solutions on the bold line between B and C are better at least in one objective function than A.

Given a (wrong) solution A, there exist (right) solutions which are better than A at least in one objective function. All the solutions corresponding to points in Fig. 1.3 which do not lie on the bold line (defined by the two end

points y_{min} and m_{min}) are *wrong solutions* to be discarded by a designer. Conversely, the good or *optimal solutions* are those and only those which are represented by points lying on the said bold line, the set of these solutions is called *Pareto-optimal set*[5].

The task of the designer is that of choosing a solution only from the Pareto-optimal set.

It is evident that the designer can choose among an unlimited number of solutions, i.e. the solution is not unique.

This is due to the fact that the addressed optimisation problem required to minimise concurrently $m(b,h)$ and $y(b,h)$, i.e. *the minimisation of a vector function minimisation of a vector function* had to be performed.

The minimisation of a vector function is an issue typical in the optimisation of complex systems. Presently, it does not seem to be very well known by designers, even if the theory was developed more than one century ago. On the contrary, the minimisation of a *scalar function*[6] of one or more variables is generally reputed as a relatively simple task.

In order to explain what does it mean 'minimising a vector function', let us start to minimise a scalar function, i.e. let us separately minimise $m(b,h)$ and $y(b,h)$. These two scalar functions are obviously to be minimised by taking into account that constraints (1.2) and (1.3), i.e. a *constrained minimisation* has to be performed.

If only one objective function was considered, e.g. $m(b,h)$, the problem could be formulated mathematically as follows:

- *Find* b *and* h
- *such that*

$$b_{min} \leq b \leq b_{max}$$
$$h_{min} \leq h \leq h_{max}$$

- *and such that*

$$\min m(b,h) = \min \rho bhl$$

- *subject to*

$$\sigma_{max} = 6\frac{Fl}{bh^2} \leq \sigma_{adm} = \sigma_E/\eta$$
$$F < F_{cr} = \frac{k_1 b^3 h}{l^2}\sqrt{\left(1 - k_2\frac{b}{h}\right)EG}$$

[5] Vilfredo Pareto (1848–1923) was an Italian engineer, who later became a famous sociologist and economist at the University of Lousanne

[6] for example, minimising only $m(b,h)$ or only $y(b,h)$)

The solution to this optimisation problem in which only one objective function is to be minimised can be denoted by

$$m_{m_{min}}(b_{m_{min}}, h_{m_{min}}), y_{m_{min}}(b_{m_{min}}, h_{m_{min}})$$

Similarly, by minimising $y(b, h)$ the solution can be written as

$$m_{y_{min}}(b_{y_{min}}, h_{y_{min}}), y_{y_{min}}(b_{y_{min}}, h_{y_{min}})$$

It has to be noticed that if a scalar function has to be minimised, the solution is unique

$$\min m(b, h) \rightarrow m_{m_{min}}(b_{m_{min}}, h_{m_{min}}), y_{m_{min}}(b_{m_{min}}, h_{m_{min}}) \rightarrow 1\,\text{solution}$$

$$\min y(b, h) \rightarrow m_{y_{min}}(b_{y_{min}}, h_{y_{min}}), y_{y_{min}}(b_{y_{min}}, h_{y_{min}}) \rightarrow 1\,\text{solution}$$

but, if a vector function has to be minimised, the solution is not unique

$$\min \begin{pmatrix} m(b, h) \\ y(b, h) \end{pmatrix} \rightarrow \text{Pareto-optimal set} \rightarrow \infty^1 \text{optimal solutions}$$

The minimisation of a vector function involves the definition of solutions. So the concurrent optimisation of more than two objective functions involves finding infinite solutions.

This occurrence is very important and may be reputed as the key issue of multi-objective programming.

The Pareto-optimal set is the set which contains all the infinite solutions coming from the minimisation of a vector function.

The designer has to select one solution from the Pareto-optimal set, and this selection is inherently somewhat subjective. The Pareto-optimal set does contain (by definition) all the best compromise solutions between conflicting objective functions. All of the Pareto-optimal solutions do have the status of the best compromise solutions. The degree or level of the compromise varies in a fuzzy way among solutions. Selecting the desired level of the compromise is the primary or ultimate task of the designer.

The designer has to act as a *decision maker*, actually, by selecting a solution from the Pareto-optimal set, he decides his desired best compromise between conflicting objective functions.

In our example, the designer has to chose (from the Pareto-optimal set, and from this set only) the preferred couple of values $m(b, h), y(b, h)$. It is clear that the designer will disregard all of the so-called dominated solutions (the solutions corresponding to points lying inside the manifold in Fig. 1.2 (b)), and will chose one solution from the dominating ones (belonging to the Pareto-set).

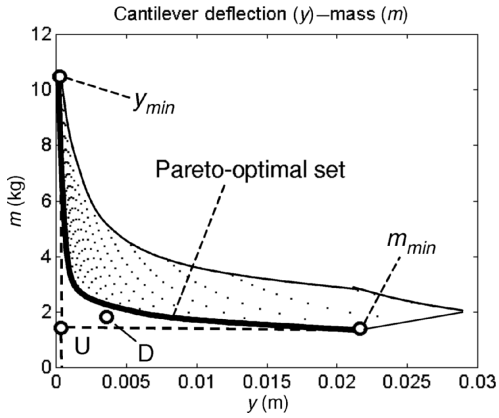

Fig. 1.4. Definition of the utopia or ideal point U (coordinates are m_{min}, y_{min}). The objective functions (m, y) referring to point D cannot be attained by the considered system. Data in Fig. 1.2

In the literature [59, 238], many attempts have been made to help the designer to make a choice. The results are often questionable, mainly because it is assumed that the designer knows perfectly the desired compromise to be reached among conflicting objective functions. This is not always the case, or better, in the design of complex systems, it is *never* the case.

For example, even in our case, it is not easy to answer the questions *how small the deflection should be* and *how light the cantilever should be*? It is a matter which might involve additional considerations which were not taken into account at the stage of problem formulation.

What actually happens during the optimisation of a complex system is that the designer, before making his choice, acquires an in–depth knowledge of the system performances under investigation. Particularly, he understands what are the limit performances of the system.

For example, in our case, he will realise that an objective function defined by point D in Fig. 1.4 will never be attained, unless the system is changed. Also y_{min} and m_{min} are important boundaries to the performances (i.e. objective functions). These boundaries, depending on the problem under investigation, could be changed by varying the design variable ranges (i.e. the ranges into which the design variables are varied during the optimisation process).

An important reference point is the *utopia or ideal point*. In our example, it is the point whose coordinates are (m_{min}, y_{min}). Obviously, a cantilever having such performances (i.e. objective functions) does not exist, however if it existed, it would be the best solution. The utopia point can be used to choose Pareto-optimal solutions. Empirical reasoning suggests to choose those Pareto-optimal solutions which are closer to the utopia point.

If the system is complex (tenth of objective functions and tenth of design variables) the knowledge and understanding process (necessary for an effective

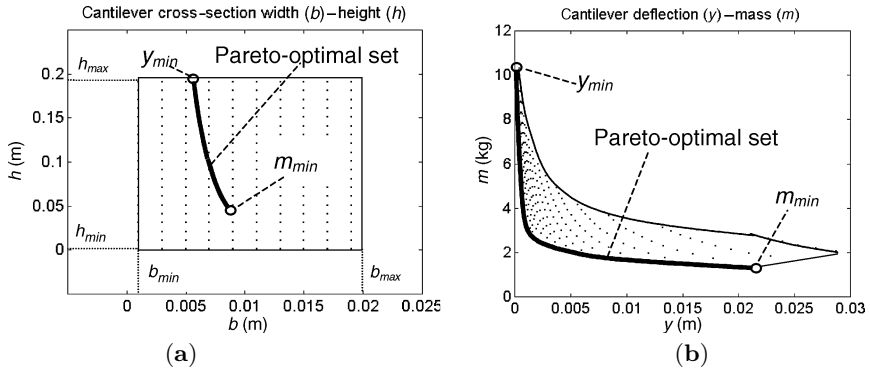

Fig. 1.5. The two Pareto-optimal sets defined, respectively, in the design variable domain (a) and in the objective function domain (b). Data in Fig. 1.2

optimisation) may be heavy and time-consuming. Sometimes the time for optimising a complex system may require as much time as that which was spent to develop the validated mathematical model of the system under investigation. When a designer has at his disposal a validated mathematical model of a system, he might be only half way (or less!) from the complete (optimal) design of the system.

As the designer has chosen the preferred compromise among conflicting objective functions by selecting a solution from the Pareto-optimal set, then the problem is to define or identify the values of the design variables related to those objective functions. In other words, the designer has to define the design variables that allow the system to perform according to his wishes.
So we have to consider that, having a set of Pareto-optimal solutions, what we really want are the values of design variables corresponding to those Pareto-optimal solutions. It can be useful to represent the Pareto-optimal solutions both in the objective functions domain or conversely, in the design variables domain (see Fig. 1.5).

With reference to our optimisation problem, in Fig. 1.5, the Pareto-optimal set defined in the design variable domain is shown. As it could be expected, the stiffer cantilever has the highest cross-section height h.
There is a direct relationship between the points of the two Pareto-optimal sets represented in Fig. 1.5. It is obvious that the designer will always select the design variables from the Pareto-optimal set represented in Fig. 1.5(a).

The shapes of the cross-sections of three Pareto-optimal cantilevers are shown in Fig. 1.6. Two of them refer respectively to points m_{min} and y_{min}. The intermediate cantilever in Fig. 1.6 has both the minimum mass m and the minimum deflection y, according to the definition of the best compromise addressed before.

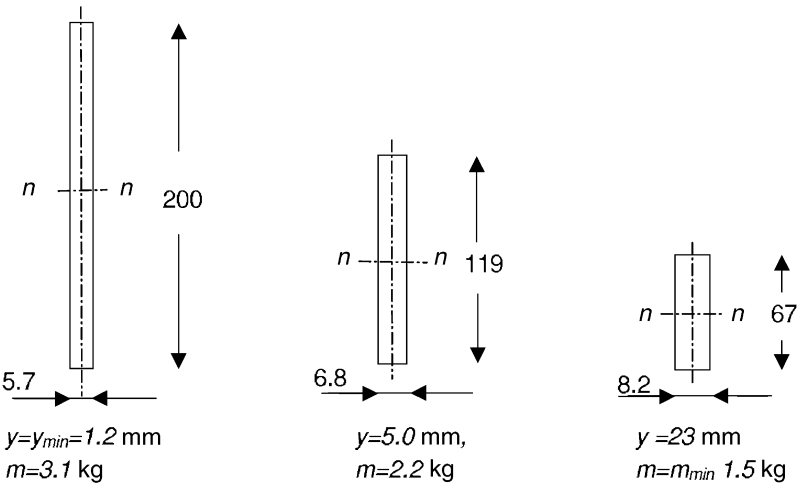

Fig. 1.6. Cross-sections of cantilevers belonging to the Pareto-optimal set. Dimensions in mm. *Left*: Minimum deflection cantilever (point y_{min} in Fig. 1.5). *Centre*: A generic Pareto-optimal cantilever. *Right*: Minimum mass cantilever (point m_{min} in Fig. 1.5). Data are shown in Fig. 1.2

1.2 Finding the Pareto-optimal Sets

From the above considerations, it is clear that a designer should always try to find the Pareto-optimal sets (both in the design variable space and in the objective function space) to obtain the best solutions. The existence of the Pareto-optimal set is stated by a theorem which guarantees non-empty Pareto-optimal sets under broad conditions[7], holding for the majority of engineering problems.

In the engineering practice, very rarely problems are solved by resorting to the computation of the Pareto-optimal sets (Fig. 1.7). In fact, the computations to find the Pareto-optimal sets are rather involved and time-consuming[8]. However, in the last years, the ever-increasing power of computers has allowed to attempt the optimal design of complex systems on the basis of Pareto theory.

1.2.1 Exhaustive Method

There are many methods to find Pareto-optimal sets (see Sect. 3.4.1). The one, called exhaustive method, which has been used in the presented example referring to the cantilever, is the simplest, but unmanageable due to the huge

[7]The assumptions are that the design variable space is closed and the objective functions are continuous functions of the design variables [166]

[8]This seems the main reason why, presently, very few designers are instructed to apply the optimisation theory based on Pareto-optimality

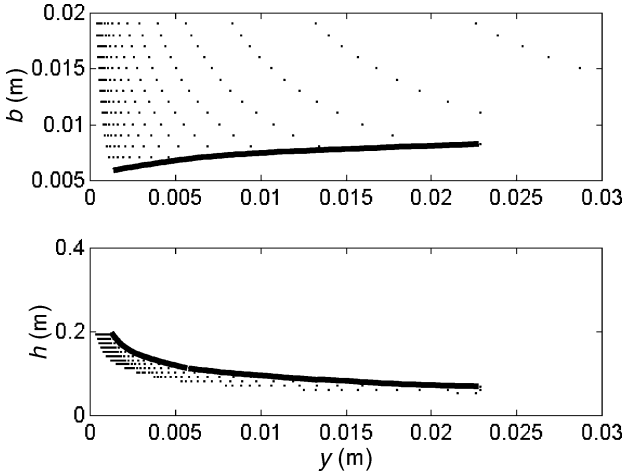

Fig. 1.7. Pareto-optimal cross-section width b and Pareto-optimal cross-section height h as function of Pareto-optimal deflection y. By these two graphs, the direct relationship between the points of the two Pareto-optimal sets in Fig. 1.5 can be obtained

amount of computations required. To explain why the exhaustive method is unmanageable, let us resort to an example.

Let us imagine that a complex system is defined by 10 design variables n_{dv} and that 30 objective functions n_{of} have to be taken into account. Each design variable may assume $n_v = 10$ different values within its definition range. Let us assume that, given a combination of design variables, the time t_s for performing a simulation to obtain the value of one single objective function is 1 s. By the exhaustive method, the number of all possible combinations of design variables values is

$$n_{cdv} = n_v^{n_{dv}} \tag{1.4}$$

and the total time t_t for computing all of the objective functions as function of all possible combinations of design variables is

$$t_t = t_s n_{of} n_v^{n_{dv}} \tag{1.5}$$

Substituting the numerical values

$$t_t = t_s n_{of} n_v^{n_{dv}} = 1 \times 30 \times 10^{10}\,\mathrm{s} = 3 \times 10^{11}\,\mathrm{s} = 9,645\,\mathrm{years} \tag{1.6}$$

the total time t_t for computing all the objective functions is clearly a time too long to solve an engineering problem (10,000 years seems to be the age of *homo sapiens sapiens*). Additionally if, for some practical reason, the number of design variables should be increased by two (from 10 to 12), the total time requested for simulations would be 100 times longer

$$t_t = t_s n_{of} n_v^{n_{dv}} = 1 \times 30 \times 10^{10+2}\,\mathrm{s} = 100 \times 9{,}645\,\mathrm{years} = 96{,}4500\,\mathrm{years} \tag{1.7}$$

These figures are startling and state evidently that an exhaustive construction[9] of the Pareto-optimal set is, in general, not possible, unless the system under consideration is very simple.

The critical factors in (1.5) are the number of design variables n_{dv}, the number of values the design variables may assume within their respective definition ranges n_v, and the simulation time for computing one single objective function t_s. Often, the number of objective functions n_{of} is not critical.

1.2.2 Uniformly Distributed Sequences and Random Search

There have been conceived many methods (see Sect. 3.4.2) to reduce the above-addressed total number of simulations ${n_v}^{n_{dv}}$, in fact, it is not absolutely necessary to explore all the possible design configurations to construct the Pareto-optimal sets.

Sometimes it is sufficient to reduce (to a considerable extent) the number of simulations to estimate properly the Pareto-optimal set both in the space of objective functions and in the space of design variables.

For example, in Fig. 1.8(a) a regular grid representing an ordered combination of design variables values is compared with two different combinations (centre and right) which are still ordered (in the sense that points are not randomly distributed) but the number of points is greatly reduced with respect to the previous combination. Points related to the combination in Fig. 1.8 (centre) are somehow 'equally' distributed along vertical planes. Every square of the three vertical planes contains one point. Points along these planes are somehow 'well distributed'. For each vertical plane, one out of the three coordinates of a point is fixed. So – in order to reconstruct the relationships between the design variables and the objective functions – the informative contribution given by the design variable, which is kept fixed, might be nearly the same for all points lying on the same plane. This causes a computational inefficiency as many simulations will be performed at a given (fixed) value of a design variable.

The distribution in Fig. 1.8(c) overcomes this problem, the information on the relationships between the objective functions and the design variables is complete and less redundant. From a mathematical point of view, the above explanation is very rough, anyway it gives some hint on how and why simulations can be reduced to estimate the Pareto-optimal sets.

Orthogonal arrays and *low discrepancy sequences*[10] are particularly suited to be used to reduce as much as possible the number of combinations of design variables used as input data for simulations. These sequences are called

[9] When all the simulations have been made on the basis of all possible combinations of design variables, the construction of all Pareto-optimal sets is performed by selecting the so-called dominating solutions from the dominated ones. This is mathematically performed by applying a proper definition that will be presented in Chap. 2

[10] Proper information on this will be given in Sect. 3.4.2

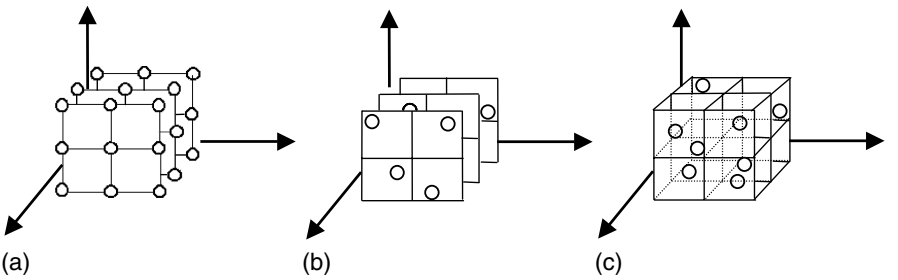

Fig. 1.8. Three different combinations of design variables values. Number of design variables: $n_{dv} = 3$. (**a**) Regular grid, number of values the design variables may assume within their respective definition ranges: $n_v = 3$, total number of design variable combinations: 27. (**b**) Low-discrepancy grid, total number of design variable combinations: 12. (**c**) Low discrepancy grid, total number of design variable combinations: 8

uniformly distributed sequences and are said to have *low discrepancy* as the points are uniformly distributed in the space. In multi-dimensional spaces (>3 up to 30 or more dimensions), the low discrepancy placement of points is a peculiar mathematical matter which requires a special theoretical background.

1.2.3 Genetic Algorithms

Another well-known method to optimise complex systems is based on Genetic algorithms (GAs)(see Sect. 3.4.4). This approach mimics what has happened during the evolution of living creatures. Living creatures evolve by adapting as much as possible to the environment. It is assumed that

- individuals have a chance to reproduce themselves according to their fitness (reproduction step),
- an individual cross its genes with the ones of its partner to generate a new individual (crossover step),
- some mutation of the genes always acts during the previous crossover process (mutation step)

The generated individuals start a new three-step generation process (reproduction, crossover, mutation). A number of generations are necessary to select fit individuals belonging to Pareto-optimal sets.

A design project evolves by adapting as much as possible both to the designer aims and to the design constraints.

An individual corresponds to a design solution, the genes correspond to a design variable combination which identifies a single design solution. A direct mathematical relationship is established between genes and design variables: the design variables values are expressed in binary form and these binary strings correspond to the genes. Each design variable can be converted into

its binary equivalent, and thereby mapped into a fixed length of zeros and ones.

An individual/design solution is selected for crossover according to its fitness. The fitness is related to the values of the objective functions. If the fitness is high, the individual/design solution will have more chances to reproduce, i.e. crossing its genes with those of a partner. The crossover process just mixes the binary strings pertaining to genes/design variables of two parents to generate one new individual. The genes/design variables of the new individual/design solution may slightly vary due to mutation. GAs do have many variants which are often developed to solve particular optimisation problems.

GAs are very useful especially when the design variables values take discrete values. Unfortunately, they are not very simple to be used (with respect to other methods such as global approximation, see next subsection) especially when dealing with the optimisation of (very) complex systems. In fact they often require the exact definition of some algorithm parameters that influence the efficiency of the search.

1.2.4 Comparison of Broadly Applicable Methods to Solve Optimisation Problems

In Table 1.1 four general and broadly applicable methods to solve optimisation problems are presented together with their optimisation properties and performances. All the four methods have been introduced in the preceding subsections. They allow a vectorial formulation, i.e. they permit the optimisation of all the objective functions concurrently (direct derivation of Pareto-optimal sets).

The computational efficiency refers to how fast an optimisation can be. For simple problems (number of design variables less than 5 or 6) all the four presented methods work well. Complex problems are handled well by GAs; on the contrary, the exhaustive method is absolutely unsuited for this kind of problems.

The accuracy in the definition of the Pareto-sets is best for the exhaustive method in which design variables may vary in non-continuous ranges (discrete values). All the four methods in Table 1.1 do have the following important properties:

- they can deal with design variables which vary within non-continuous ranges (discrete values)
- objective functions can be non-continuous functions of the design variables.

1.2.5 Global Approximation

The multi-objective optimisation of very complex systems may require still a prohibitive simulation effort, even reducing the number of combinations

Table 1.1. General and broadly applicable methods for solving optimisation problems

	Computational efficiency (time)		*Accuracy*		*Discrete design variables values allowed*	*Objective functions need to be continuous functions*
	Simple problem	*Complex problem*				
Exhaustive	−	− −	++ (*discrete design variables*)	− (*continuous design variables values*)	yes	no
Uniformly distributed Sequences	+	−	−		yes	no
Genetic algorithms	+	+	+		yes	no

Other methods, very efficient for specific cases, are reported in Table 1.3 (− −: very bad; −: bad; +: good; + +: very good).

of design variables by means of the mentioned techniques. To optimise the performances of complex systems, the relationship between design variables and objective functions can be *approximated* by means of a pure mathematical model (see Chap. 4). The parameters of the purely mathematical model (which can be either an artificial neural network (ANN) or a piecewise quadratic function, or others) are defined on the basis of a limited number of simulations performed by means of the originally validated mathematical model of the system under consideration.

In other words, there are two models, the first which is the original validated mathematical model that the designer uses to simulate the actual physical behaviour of the system to be optimised, and the second mathematical model which is a purely mathematical model, able to approximate the outputs of the first model (Fig. 1.9).

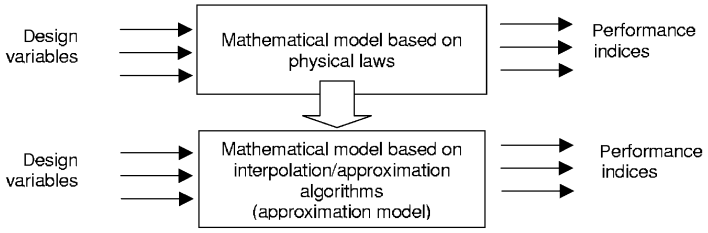

Fig. 1.9. Global approximation approach to solve optimisation problems. The original model based on physical laws is substituted by another purely mathematical model based on interpolation/approximation algorithms

Obviously it is supposed, as it always happens in actual applications, that the first model requires time-consuming simulations and the second model provides quick simulation outputs. Typically the simulation time of the second model is a small fraction (1/10 – 1/10,000) of the simulation time requested by the first model.

The accuracy in the approximation of the outputs depends on the approximation model employed. In Table 1.2, known methods for global approximation are presented. Linear and quadratic interpolation are well suited to approximate locally, i.e. in the neighbourhood of a single point, the objective functions. They are suited for very simple optimisation problems, and the tuning (i.e. the process for defining the parameters of the pure mathematical model) is relatively easy. The evaluation accuracy, i.e. the ability to reproduce the original values of objective functions given the values of design variables, is not very high. Response surface methodologies use linear and quadratic approximation models.

Radial basis functions neural networks seem to be accurate, easy to be tuned with appropriate algorithms (as will be exposed in Chap. 4) and efficient in the evaluation of objective functions. Multi-layer perceptron neural

Table 1.2. Approximation methods for solving optimization problems (–: bad; +: good; + +: very good)

	Approximation domain	*Evaluation accuracy*		*Tuning effort*
		Simple model	*Complex model*	
Linear interpolation	local	+	−	++
Quadratic interpolation	local	+	−	++
Radial basis functions Neural networks	local/global	++	+	+
Multi–layer perceptron neural networks	global	++	++	−
Statistical approximation	global	++	+	−

networks perform better than previous radial basis functions neural networks but require more effort in tuning. These approximation methods are particulary suited for complex design optimisation problems.

1.2.6 Multi-objective Programming via Non-linear Programming

Historically, multi-objective optimisation problems were numerically solved resorting to non-linear programming (NLP), i.e. by transforming the original vector problem into a scalar one (see Sect. 3.4.5). Obviously this transformation is still effective but it is not recommended any longer for solving general complex optimisation problems, i.e. finding Pareto-optimal sets.

A well-known method for transforming a vector optimisation problem into a scalar one is the *constraints method* (see Sect. 3.4.10). Given the multi-objective optimisation problem as a vector function to be minimised

$$\min \mathbf{f}(\mathbf{x}) = \min \begin{Bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \dots \\ f_{n_{of}}(\mathbf{x}) \end{Bmatrix} \qquad \mathbf{x} = \begin{Bmatrix} x_1 \\ x_2 \\ \dots \\ x_{n_{dv}} \end{Bmatrix} \tag{1.8}$$

subject to n_c constraints

$$\mathbf{g}(\mathbf{x}) = \begin{Bmatrix} g_1(\mathbf{x}) \\ g_2(\mathbf{x}) \\ \dots \\ g_{n_c}(\mathbf{x}) \end{Bmatrix} \leq 0 \tag{1.9}$$

the original vector problem is transformed into the new scalar one

$$\min f_1(\mathbf{x}) \qquad \mathbf{x} = \begin{Bmatrix} x_1 \\ x_2 \\ \dots \\ x_{n_{dv}} \end{Bmatrix} \tag{1.10}$$

subject to the $n_c + (n_{of} - 1)$ constraints

$$\mathbf{g}(\mathbf{x}) = \left\{ \begin{array}{c} f_2(\mathbf{x}) - \varepsilon_2 \\ \dots \\ f_{n_{of}}(\mathbf{x}) - \varepsilon_{n_{of}} \\ g_1(\mathbf{x}) \\ g_2(\mathbf{x}) \\ \dots \\ g_{n_c}(\mathbf{x}) \end{array} \right\} \leq 0 \tag{1.11}$$

By varying the values of the elements of the vector

$$\varepsilon = \left\{ \begin{array}{c} \varepsilon_2 \\ \dots \\ \varepsilon_{n_{of}} \end{array} \right\} \leq 0 \tag{1.12}$$

the Pareto-optimal set can be found.

This scalarisation can be applied to any kind of optimisation problem. It is not straightforward how to vary ε, so this method is only used to refine a Pareto-optimal solution. In other words, if a Pareto-optimal solution is known approximately, by applying the constraints method in the neighbourhood of the approximately known solution, the approximation can be significantly improved to the desired extent.

Another widespread method for scalarisation is the weight method (see Sect. 3.4.9). Given the multi-objective optimisation problem as a vector function to be minimised

$$\min \mathbf{f}(\mathbf{x}) = \min \left\{ \begin{array}{c} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \dots \\ f_{n_{of}}(\mathbf{x}) \end{array} \right\} \qquad \mathbf{x} = \left\{ \begin{array}{c} x_1 \\ x_2 \\ \dots \\ x_{n_{dv}} \end{array} \right\} \tag{1.13}$$

subject to n_c constraints

$$\mathbf{g}(\mathbf{x}) = \left\{ \begin{array}{c} g_1(\mathbf{x}) \\ g_2(\mathbf{x}) \\ \dots \\ g_{n_c}(\mathbf{x}) \end{array} \right\} \leq 0 \tag{1.14}$$

the original vector problem is transformed into the new scalar one

$$\min(w_1 f_1(\mathbf{x}) + w_2 f_2(\mathbf{x}) + \ldots + w_{n_{of}} f_{n_{of}}(\mathbf{x})) \qquad \mathbf{x} = \left\{ \begin{array}{c} x_1 \\ x_2 \\ \dots \\ x_{n_{dv}} \end{array} \right\} \tag{1.15}$$

subject to the $n_c + (n_{of} - 1)$ constraints

$$\mathbf{g}(\mathbf{x}) = \left\{ \begin{array}{c} g_1(\mathbf{x}) \\ g_2(\mathbf{x}) \\ \dots \\ g_{n_c}(\mathbf{x}) \end{array} \right\} \leq 0 \tag{1.16}$$

by varying the elements of the vector

$$\mathbf{w} = \left\{ \begin{array}{c} w_1 \\ w_2 \\ \dots \\ w_{n_{of}} \end{array} \right\} \tag{1.17}$$

the Pareto-optimal set can be found both in the space of objective functions and in the space of design variables, provided that the Hessian of $w_1 f_1(\mathbf{x}) + w_2 f_2(\mathbf{x}) + \ldots + w_{n_{of}} f_{n_{of}}(\mathbf{x})$ is a semi-definite positive matrix in the design variable space (a sufficient condition [166] for this is that $w_1 f_1(\mathbf{x}) + w_2 f_2(\mathbf{x}) + \ldots + w_{n_{of}} f_{n_{of}}(\mathbf{x})$ is globally convex in the design variable space).

This scalarisation can be applied to any kind of convex optimisation problem, for other problems it might fail in finding all Pareto-optimal solutions.

1.2.7 Algorithms to Solve Optimisation Problems in Scalar Form

In Table 1.3, the more used algorithms for solving multi-objective optimisation problems via scalar formulation are presented and compared. Solving the optimisation problem in scalar form deals with the minimisation of one function of many design variables subject to constraints (on design variables) (see Chap. 3).

Obviously, the algorithms that could solve the optimisation problem in vector form are still available for solving the scalar problem.

The *simplex algorithm* performs well and does not require the computation of the derivatives of the function.

Sequential unconstrained minimisation technique (SUMT) is a kind of gradient method and requires the function to be continuous together with its first derivative. Discrete values are not permitted because the search is based on gradient evaluation.

Sequential quadratic programming (SQP) performs better than SUMT, and requires the same conditions on the functions to be minimised.

The main disadvantage of these methods are that they are all local methods. So the finding of global minima are all influenced by the starting point that must be close to the global solution.

1.3 Understanding Pareto-optimal Solutions

After Pareto-optimal sets have been computed, the designer can make a choice and select from these sets the preferred solution featuring the desired compromise among objective functions.

Table 1.3. Methods for solving multi-objective optimisation problems via non-linear programming (scalar formulation) (– –: very bad; –: bad; +: good; + +: very good)

	Computational efficiency		*Accuracy*		*Discrete design variables values allowed*	*Objective functions continuous functions*
	Simple problem	*Complex problem*				
Exhaustive	–	– –	++ (*discrete design variables values*)	– (*continuous design variables values*)	yes	no
Uniformly distributed sequences	+	–	–		yes	no
Genetic algorithms	+	+	+		yes	no
Simplex	+	–	+		yes	no
Sequential unconstrained minimisation technique	+	–	+		no	yes
Sequential quadratic programming	+	+	+		no	yes

This process can be preceded and even accelerated by special analyses allowing the designer to have an insight into the physical phenomena he is trying to analyse (see Sect. 3.3.3). In other words there are some analyses by which the designer may understand the reason why a Pareto-optimal solution requires such a design variable combination. In particular, these analyses may show which is

- the relationship between two Pareto-optimal objective functions,
- the relationship between two Pareto-optimal design variables,
- the relationship between a Pareto-optimal design variable and a Pareto-optimal objective function.

In Fig. 1.10 these different relationships are shown. The Pareto-optimal set is defined in an n_{of}-dimensional domain, so the Pareto-optimal values projected onto a bidimensional domain (f_i, f_j) appear in a picture like the one in Fig. 1.10. Thus, the addressed analyses can be performed on the basis of a statistical approach possibly combined with the plotting of graphs like those represented in Fig. 1.10. The *Spearman rank–order correlation coefficient* can be used to assess the above-introduced relationships and thus is particularly suited to discover significant relationships. This coefficient is able to identify monotonic non-linear relationships and thus is particularly suited to discover significant relationships that may suggest important engineering solutions. If

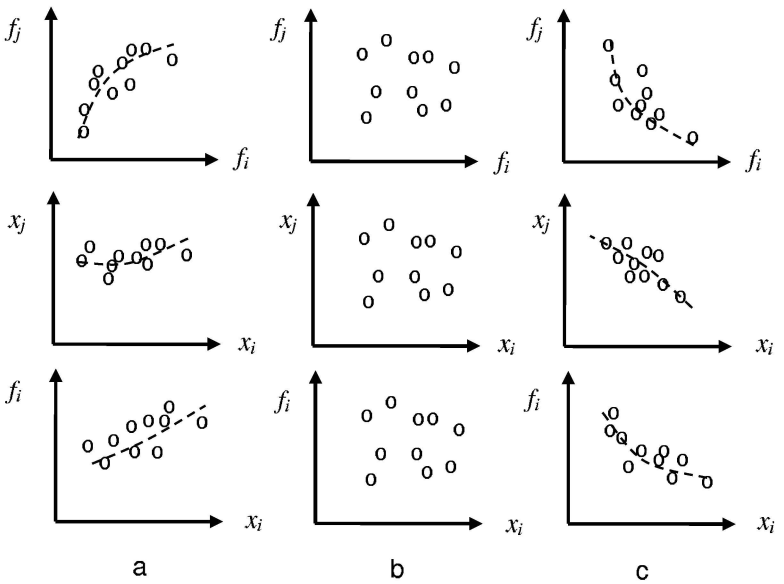

Fig. 1.10. Relationships between Pareto-optimal objective functions (f_i–f_j), between Pareto-optimal design variables (x_i–x_j), and between Pareto-optimal objective functions and Pareto-optimal design variable (f_i–x_i). The values can be either directly correlated (**a**), uncorrelated (**b**), or indirectly correlated (**c**)

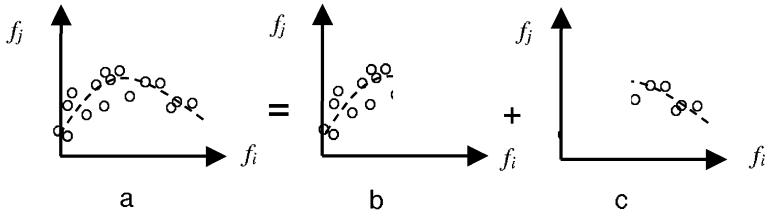

Fig. 1.11. Partition of the f_i domain to obtain two monotonic interpolation functions f_i. The Spearman rank–order correlation coefficient can be computed separately for cases (**b**) and (**c**)

the relationships are not monotonic a partition of the domain could be performed as shown in Fig. 1.11.

By compusting the Spearman rank–order correlation coefficient, very useful information on the system under consideration can be gained. In fact, a kind of sensitivity analysis focused on the Pareto-optimal solutions can be performed. In conventional sensitivity analysis, the relationships between objective functions and design variables are computed in the neighbourhood of one simple reference design solution. The analysis does not distinguish dominated from non-dominated solutions, i.e. the analysis is *not* restricted to Pareto-optimal solutions. So very poor information can be gained with conventional parameter sensitivity analysis. An analysis restricted to the very special Pareto-optimal values only may give significant hints to the designer. Actually, the relationships between Pareto-optimal objective functions and Pareto-optimal design variables can be highlighted.