Chapter 1
Introduction

Patrick Siarry

Every day, engineers and decision-makers are confronted with problems of growing complexity in diverse technical sectors, for example in operations research, the design of mechanical systems, image processing, and, particularly, electronics (CAD of electrical circuits, the placement and routing of components, improvement of the performance or manufacturing yield of circuits, characterization of equivalent schemas, training of fuzzy rule bases or neural networks, …). The problem to be solved can often be expressed as an *optimization problem*. Here one defines an objective function (or several such functions), or cost function, which one seeks to minimize or maximize vis-à-vis all the parameters concerned. The definition of the optimization problem is often supplemented by information in the form of *constraints*. All the parameters of the solutions adopted must satisfy these constraints, otherwise these solutions are not realizable. In this book, our interest is focused on a group of methods, called *metaheuristics* or meta-heuristics, which include in particular the simulated annealing method, evolutionary algorithms, the tabu search method, and ant colony algorithms. These have been available from the 1980s and have a common aim: to solve the problems known as *hard optimization* as well as possible.

We will see that metaheuristics are largely based on a common set of principles which make it possible to design solution algorithms; the various regroupings of these principles thus lead to a large variety of metaheuristics.

P. Siarry (✉)
Laboratoire Images, Signaux et Systèmes Intelligents (LiSSi, E.A. 3956),
Université Paris-Est Créteil Val-de-Marne,
122 rue Paul Armangot, 94400 Vitry-sur-Seine, France
e-mail: siarry@u-pec.fr

P. Siarry (ed.), *Metaheuristics*, DOI 10.1007/978-3-319-45403-0_1

1.1 "Hard" Optimization

Two types of optimization problems can be distinguished: "discrete" problems and problems with continuous variables. To be more precise, let us quote two examples. Among the discrete problems, one can discuss the well-known traveling salesman problem: this is a question of minimizing the length of the route of a "traveling salesman," who must visit a certain number of cities before return to the town of departure. A traditional example of a continuous problem is that of a search for the values to be assigned to the parameters of a numerical model of a process, so that the model reproduces the real behavior observed as accurately as possible. In practice, one may also encounter "mixed problems," which comprise simultaneously discrete variables and continuous variables.

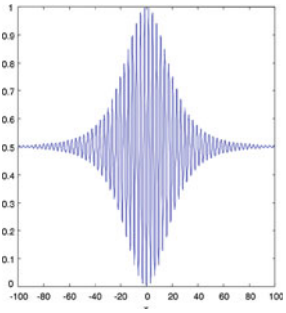
This differentiation is necessary to determine the domain of hard optimization. In fact, two kinds of problems are referred to in the literature as hard optimization problems (this name is not strictly defined and is bound up with the state of the art in optimization):

- Certain discrete optimization problems, for which there is no knowledge of an exact *polynomial* algorithm (i.e., one whose computing time is proportional to N^n, where N is the number of unknown parameters of the problem and n is an integer constant). This is the case, in particular, for the problems known as "NP-hard," for which it has been conjectured that there is no constant n for which the solution time is limited by a polynomial of degree n.
- Certain optimization problems with continuous variables, for which there is no knowledge of an algorithm that enables one to definitely locate a global optimum (i.e., the best possible solution) in a finite number of computations.

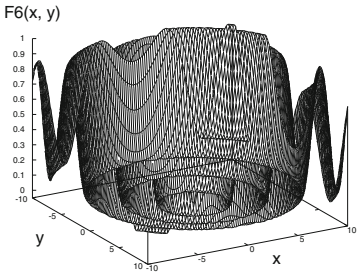
Many efforts have been made for a long time, separately, to solve these two types of problems. In the field of continuous optimization, there is thus a significant arsenal of traditional methods for *global optimization* [1], but these techniques are often ineffective if the objective function does not possess a particular structural property, such as convexity. In the field of discrete optimization, a great number of *heuristics*, which produce solutions close to the optimum, have been developed; but the majority of them were conceived specifically for a given problem.

The arrival of *metaheuristics* marks a reconciliation of the two domains: indeed, they can be applied to all kinds of discrete problems and they can also be adapted to continuous problems. Moreover, these methods have in common the following characteristics:

- They are, at least to some extent, *stochastic*: this approach makes it possible to counter the *combinatorial explosion* of the possibilities.
- They are generally of discrete origin, and have the advantage, decisive in the continuous case, of being direct, i.e., they do not resort to often problematic calculations of the gradients of the objective function.



(a) One-dimensional representation in the domain [−100, 100]



(b) Two-dimensional representation in the domain [−10, 10]

Fig. 1.1 Shape of the test function F6

- They are inspired by *analogies*: with physics (simulated annealing, simulated diffusion, …), with biology (evolutionary algorithms, tabu search, …), or with ethology (ant colonies, particle swarms, …).
- They also share the same disadvantages: difficulties of *adjustment* of the parameters of the method, and a *large computation time*.

These methods are not mutually exclusive: indeed, with the current state of research, it is generally impossible to envisage with certainty the effectiveness of a given method when it is applied to a given problem. Moreover, the current tendency is the emergence of *hybrid methods*, which endeavor to benefit from the specific advantages of different approaches by combining them. Finally, another aspect of the richness of metaheuristics is that they lend themselves to all kinds of *extensions*. We can quote, in particular:

- *multiobjective* optimization [6], which is a question of optimizing several contradictory objectives simultaneously;
- *multimodal* optimization, where one endeavors to locate a whole set of global or local optima;
- *dynamic* optimization, which deals with temporal variations of the objective function;
- the use of *parallel implementations*.

These extensions require, for the development of solution methods, various specific properties which are not present in all metaheuristics. We will reconsider this subject, which offers a means for guiding the user in the choice of a metaheuristic, later. The adjustment and comparison of metaheuristics are often carried out empirically, by exploiting analytical sets of test functions whose global and local minima are known. We present in Fig. 1.1 the shape of one of these test functions as an example.

1.2 Source of the Effectiveness of Metaheuristics

To facilitate the discussion, let us consider a simple example of an optimization problem: that of the placement of the components of an electronic circuit. The objective function to be minimized is the length of the connections, and the unknown factors—called "decision variables"—are the sites of the circuit components. The shape of the objective function of this problem can be represented schematically as in Fig. 1.2, according to the "configuration": each configuration is a particular placement, associated with a choice of a value for each decision variable. Throughout the entire book—except where otherwise explicitly mentioned—we will seek in a similar way to minimize an objective. When the space of the possible configurations has such a tortuous structure, it is difficult to locate the global minimum c^*. We explain below the failure of a "classical" iterative algorithm, before commenting on the advantage that we can gain by employing a metaheuristic.

1.2.1 Trapping of a "Classical" Iterative Algorithm in a Local Minimum

The principle of a traditional "iterative improvement" algorithm is the following: one starts from an initial configuration c_0, which can be selected at random, or—for example in the case of the placement of an electronic circuit—can be determined by a designer. An elementary modification is then tested; this is often called a "movement" (for example, two components chosen at random are swapped, or one of them is relocated). The values of the objective function are then compared, before and after this modification. If the change leads to a reduction in the objective function, it is accepted, and the configuration c_1 obtained, which is a "neighbor" of the preceding one, is used as the starting point for a new test. In the opposite case, one returns to the preceding configuration before making another attempt. The process is carried out iteratively until any modification makes the result worse. Figure 1.2 shows that this algorithm of iterative improvement (also known as the *classical method* or *descent method*) does not lead, in general, to the global optimum, but only to one

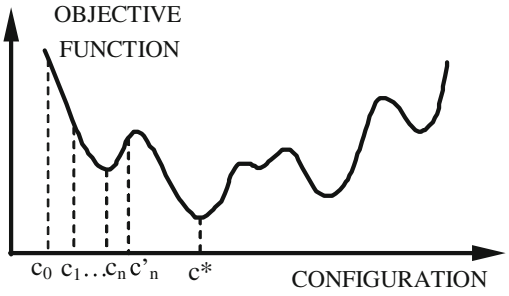


Fig. 1.2 Shape of the objective function of a hard optimization problem depending on to the "configuration"

local minimum c_n, which constitutes the best accessible solution taking the initial assumption into account.

To improve the effectiveness of the method, one can of course apply it several times, with arbitrarily selected different initial conditions, and retain as the final solution the best local minimum obtained. However, this procedure appreciably increases the computing time of the algorithm, and may not find the optimal configuration c^*. The repeated application of the descent method does not guarantee its termination and it is particularly ineffective when the number of local minima grows exponentially with the size of the problem.

1.2.2 Capability of Metaheuristics to Extract Themselves from a Local Minimum

Another idea for overcoming the obstacle of local minima has been demonstrated to be very profitable, so much so that it is the basic core of all metaheuristics based on a *neighborhood* (the simulated annealing and tabu methods). This is a question of authorizing, from time to time, movements of *increase*, in other words, accepting a temporary degradation of the situation, during a change in the current configuration. This happens, for example, if one passes from c_n to c'_n in Fig. 1.2. A mechanism for controlling these degradations—specific to each metaheuristic—makes it possible to avoid divergence of the process. It consequently becomes possible to extract the process from a trap which represents a local minimum, to allow it to explore another more promising "valley." The "distributed" metaheuristics (such as evolutionary algorithms) also have mechanisms allowing the departure of a particular solution out of a local "well" of the objective function. These mechanisms (such as *mutation* in evolutionary algorithms) affect the solution in hand; in this case, they help the collective mechanism for fighting against local minima, represented by the parallel control of a "population" of solutions.

1.3 Principles of the Most Widely Used Metaheuristics

1.3.1 Simulated Annealing

Kirkpatrick and his colleagues were specialists in statistical physics, who were interested specifically in the low-energy configurations of disordered magnetic materials, referred to by the term *spin glasses*. The numerical determination of these configurations posed frightening problems of optimization, because the "energy landscape" of a spin glass contains several "valleys" of unequal depth; it is similar to the "landscape" in Fig. 1.2. Kirkpatrick et al. [14] (and, independently, Cerny [2]) proposed to deal with these problems by taking as a starting point the experimental technique

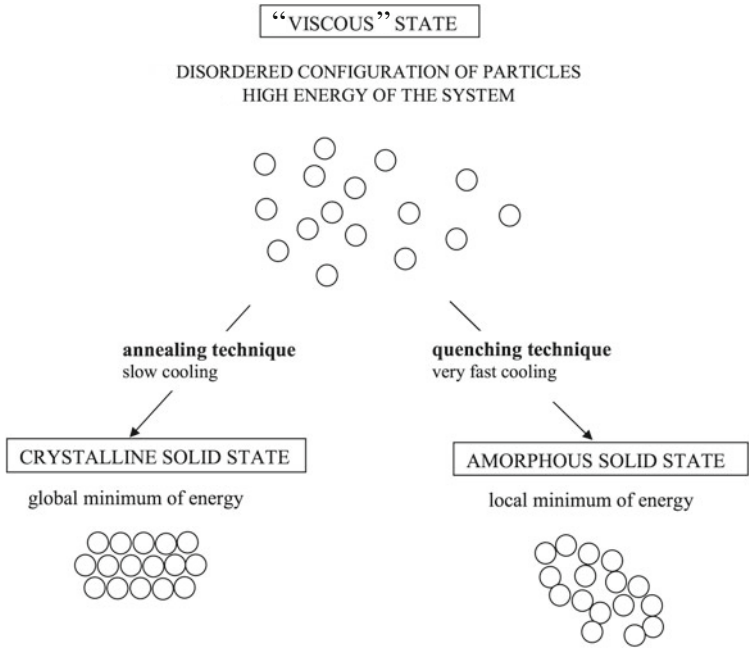


Fig. 1.3 Comparison of the techniques of annealing and quenching

of *annealing* used by metallurgists to obtain a "well-ordered" solid state, of minimum energy (avoiding the "metastable" structures characteristic of local minima of the energy). This technique consists in heating a material to a high temperature and then lowering this temperature slowly. To illustrate the phenomenon, we represent in Fig. 1.3 the effect of the annealing technique and that of the opposite technique of *quenching* on a system consisting of a set of particles.

The simulated annealing method transposes the process of annealing to the solution of an optimization problem: the objective function of the problem, similarly to the energy of a material, is then minimized, with the help of the introduction of a fictitious *temperature*, which in this case is a simple controllable parameter of the algorithm.

In practice, the technique exploits the Metropolis algorithm, which enables us to describe the behavior of a thermodynamic system in "equilibrium" at a certain temperature. On the basis of a given configuration (for example, an initial placement of all the components), the system is subjected to an elementary modification (for example, one may relocate a component or swap two components). If this transformation causes the objective function (or *energy*) of the system to decrease, it is accepted. On the other hand, if it causes an increase ΔE in the objective function, it can also be accepted, but with a probability $e^{\frac{-\Delta E}{T}}$. This process is then repeated in an iterative manner, keeping the temperature constant, until thermodynamic equilibrium is reached, at the end of a "sufficient" number of modifications. Then the

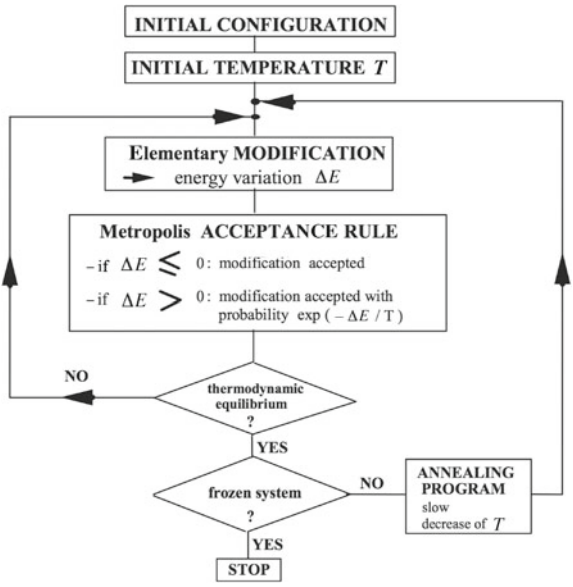


Fig. 1.4 Flowchart of the simulated annealing algorithm

temperature is lowered, before implementing a new series of transformations: the rule by which the temperature is decreased in stages is often empirical, just like the criterion for program termination.

A flowchart of the simulated annealing algorithm is schematically presented in Fig. 1.4. When it is applied to the problem of the placement of components, simulated annealing generates a disorder–order transformation, which is represented in pictorial manner in Fig. 1.5. One can also visualize some stages of this ordering by applying the method of placement of components to the nodes of a grid (see Fig. 1.6).

The disadvantages of simulated annealing lie in the "adjustments," such as the management of the decrease in the temperature; the user must have the know-how about "good" adjustments. In addition, the computing time can become very significant, which has led to parallel implementations of the method. On the other hand, the simulated annealing method has the advantage of being flexible with respect to the evolution of the problem and easy to implement. It has given excellent results for a number of problems, generally of big size.

1.3.2 The Tabu Search Method

The method of searching with tabus, or simply the *tabu search* or *tabu method*, was formalized in 1986 by Glover [10]. Its principal characteristic is based on the use of mechanisms inspired by human memory. The tabu method, from this point of view, takes a path opposite to that of simulated annealing, which does not utilize memory

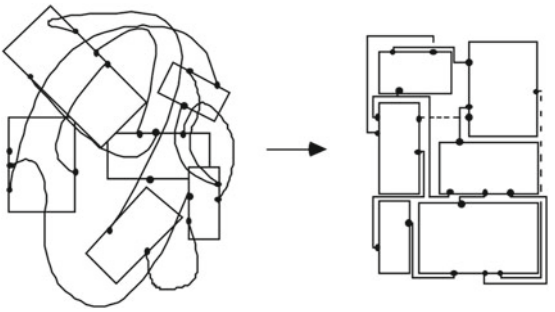


Fig. 1.5 Disorder–order transformation created by simulated annealing applied to the placement of electronic components

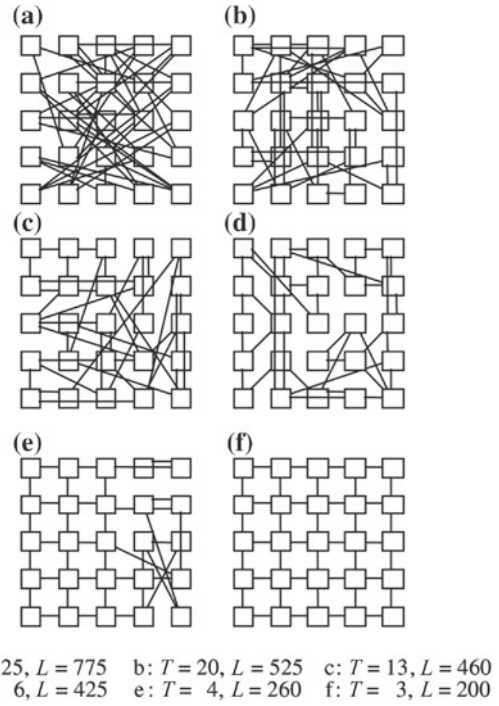


Fig. 1.6 Evolution of a system at various temperatures, on the basis of an arbitrary configuration: L indicates the overall length of the connections

at all, and thus is incapable of learning lessons from the past. On the other hand, the modeling of memory introduces multiple degrees of freedom, which hinders—even in the opinion of the original author [11]—any rigorous mathematical analysis of the tabu method. The guiding principle of the tabu method is simple: like simulated annealing, the tabu method functions at any given time with only one "current configuration" (at the beginning, an unspecified solution), which is updated during successive "iterations." In each iteration, the mechanism of passage of a configuration, called s, to the next one, called t, comprises two stages:

- One builds the set of *neighbors* of s, i.e., the set of the configurations that are accessible in only one elementary movement of s (if this set is too large, one applies a technique for reduction of its size: for example, one may utilize a list of candidates, or extract at random a subset of neighbors of fixed size). Let $V(s)$ be the set (or a subset) of these neighbors.
- One evaluates the objective function f of the problem for each configuration belonging to $V(s)$. The configuration t which succeeds s in the series of solutions built by the tabu method is the configuration of $V(s)$ in which f takes the minimum value. Note that this configuration t is adopted even if it is worse than s, i.e., if $f(t) > f(s)$: this characteristic helps the tabu method to avoid the trapping of f in local minima.

The procedure cannot be used precisely as described above, because there is a significant risk of returning to a configuration already obtained in a preceding iteration, which generates a cycle. To avoid this phenomenon, the procedure requires the updating and exploitation, in each iteration, of a list of prohibited movements, the "tabu list." This list—which gave its name to the method—contains m movements ($t \rightarrow s$), which are the opposite of the last m movements ($s \rightarrow t$) carried out. A flowchart of this algorithm, known as the"simple tabu," is represented Fig. 1.7.

The algorithm thus models a rudimentary form of memory, a *short-term memory* of the solutions visited recently. Two additional mechanisms, named *intensification* and *diversification*, are often implemented to equip the algorithm with a *long-term memory* also. This process does not exploit the temporal proximity of particular events more, but rather the frequency of their occurrence over a longer period. Intensification consists in looking further into the exploration of certain areas of the solution space, identified as particularly promising ones. Diversification is, in contrast, the periodic reorientation of the search for an optimum towards areas seldom visited until now.

For certain optimization problems, the tabu method has given excellent results; moreover, in its basic form, the method has fewer adjustable parameters than simulated annealing, which makes it easier to use. However, the various additional mechanisms, such as intensification and diversification, bring a notable amount of complexity with them.

1.3.3 Genetic Algorithms and Evolutionary Algorithms

Evolutionary algorithms (EAs) are search techniques inspired by the biological evolution of species and appeared at the end of the 1950s [9]. Among several approaches [8, 13, 16], genetic algorithms (GAs) constitute certainly the most well-known example, following the publication in 1989 of the well-known book by Goldberg [12]. Evolutionary methods initially aroused limited interest, because of their significant cost of execution. But, in the last ten years, they have experienced considerable development, which can be attributed to the significant increase in the computing power

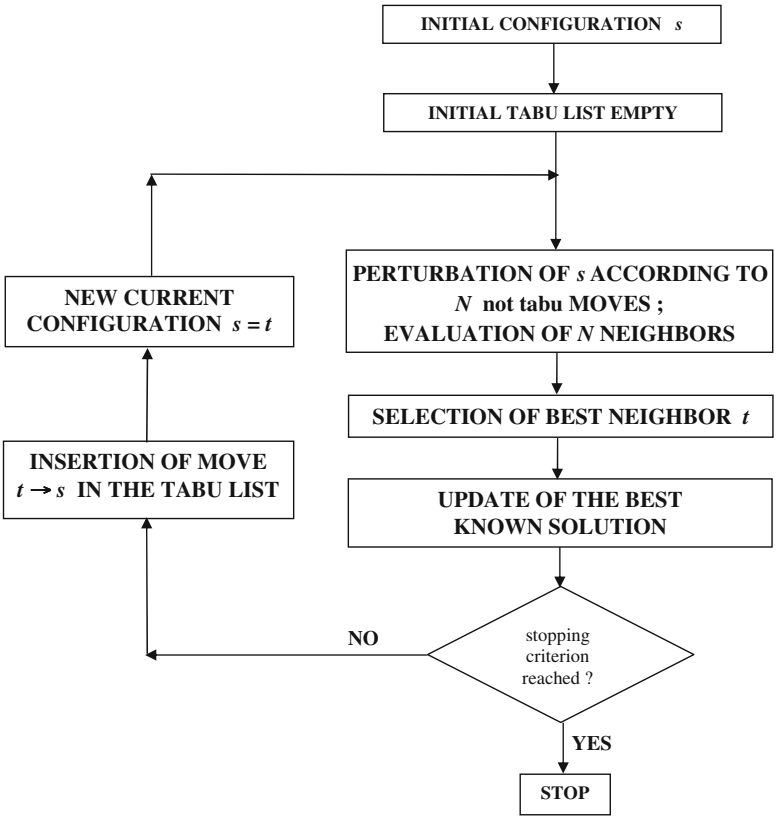
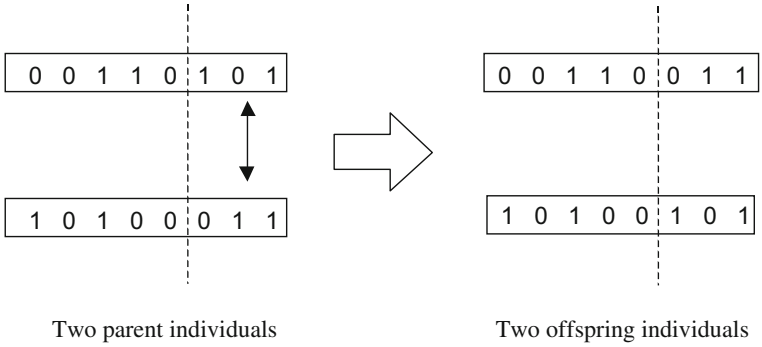


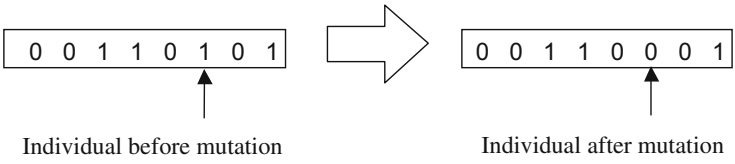
Fig. 1.7 Flowchart of the simple tabu algorithm

of computers, in particular, following the appearance of massively parallel architectures, which exploit "intrinsic parallelism" (see for example [5] for an application to the placement of components). The principle of a simple evolutionary algorithm can be described as follows: a set of N points in a search space, selected a priori at random, constitutes the *initial population*; each individual x of the population has a certain performance, which measures its degree of *adaptation* to the objective aimed at. In the case of the minimization of an objective function f, x becomes more powerful as $f(x)$ becomes smaller. An EA consists in evolving gradually, in successive *generations*, the composition of the population, with its size being kept constant. During generations, the objective is to improve overall the performance of the individuals. One tries to obtain such a result by mimicking the two principal mechanisms which govern the evolution of living beings according to Darwin's theory:

- *selection*, which favors the reproduction and survival of the fittest individuals;
- *reproduction*, which allows mixing, recombination and variation of the hereditary features of the parents, to form descendants with new potentialities.



(a) Crossover (one-point)



(b) Mutation (one single bit)

Fig. 1.8 Examples of crossover and mutation operators, in the case of individuals represented by binary strings of eight numbers

In practice, a representation must be selected for the individuals of a population. Classically, an individual can be a list of integers for a combinatorial problem, a vector of real numbers for a numerical problem in a continuous space, or a string of binary numbers for a Boolean problem; one can even combine these representations into complex structures if the need is so felt. The passage from one generation to the next proceeds in four phases: a phase of selection, a phase of reproduction (or variation), a phase of performance evaluation, and a phase of replacement. The selection phase designates the individuals that take part in reproduction. They are chosen, possibly on several occasions, a priori more often the powerful, they are. The selected individuals are then available for the reproduction phase. This phase consists in applying variation operators to copies of the individuals selected to generate new individuals; the operators most often used are *crossover* (or *recombination*), which produces one or two descendants starting from two parents, and *mutation*, which produces a new individual starting from only one individual (see Fig. 1.8 for an example). The structure of the variation operators depends largely on the representation selected for the individuals. The performances of the new individuals are then evaluated during the evaluation phase, starting from the objectives specified. Lastly, the replacement phase consists in choosing the members of the new generation: one can, for example, replace the least powerful individuals of the population

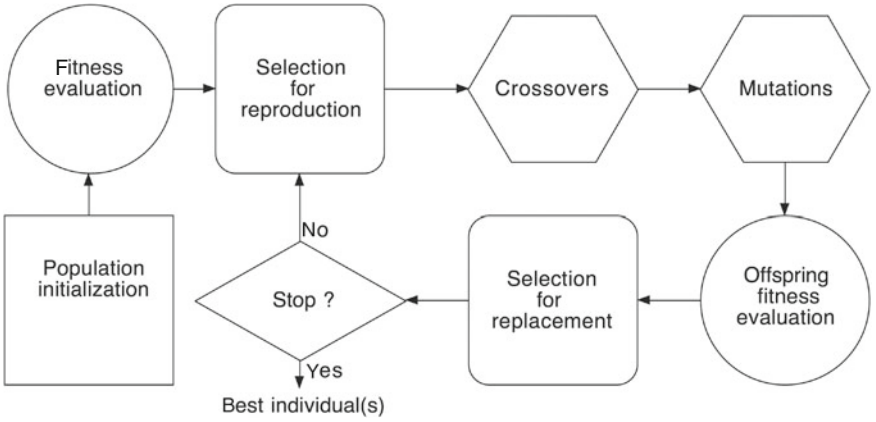


Fig. 1.9 Principle of an evolutionary algorithm

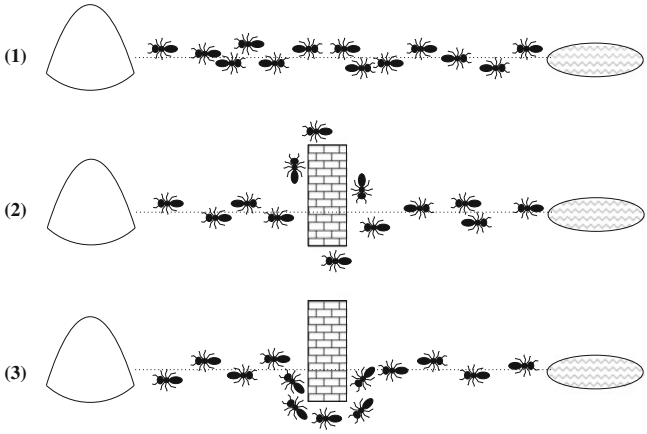
by the best individuals produced, in equal numbers. The algorithm is terminated after a certain number of generations, according to a termination criterion to be specified. Figure 1.9 represents the principle of an evolutionary algorithm.

Because they handle a population of solution instances, evolutionary algorithms are particularly suitable for finding a set of different solutions when an objective function has several global optima. In this case they can provide a sample of compromise solutions when one is solving problems with several objectives, possibly contradictory. These possibilities are discussed more specifically in Chap. 11.

1.3.4 Ant Colony Algorithms

This approach, proposed by Colorni et al. [7], endeavors to simulate the collective capability to solve certain problems observed in colonies of ants, whose members are individually equipped with very limited faculties. Ants came into existence on earth over 100 million years ago and they are one of the most successful species: 10 million billion individuals, living everywhere on the planet. Their total weight is of the same order of magnitude as that of humans! Their success raises many questions. In particular, entomologists have analyzed the collaboration which occurs between ants seeking food outside an anthill. It is remarkable that the ants always follow the same path, and this path is the shortest possible one. This is the result of a mode of indirect communication via the environment called "stigmergy." Each ant deposits along its path a chemical substance, called a pheromone. All members of the colony perceive this substance and direct their walk preferentially towards the more "odorous" areas.

This results particularly in a collective ability to find the shortest path quickly after the original path has been blocked by an obstacle (Fig. 1.10). Although this



1. Real ants follow a path between the nest and a source of food.
2. An obstacle appears on the path, and the ants choose to turn to the left or right with equal probabilities; the pheromone is deposited more quickly on the shortest path.
3. All the ants choose the shortest path.

Fig. 1.10 Ability of an ant colony to find the shortest path after the path has been blocked by an obstacle

behavior has been taken as a starting point for modeling the algorithm, Colorni et al. [7] proposed a new algorithm for the solution of the traveling salesman problem. Since this research work, the method has been extended to many other optimization problems, some combinatorial and some continuous.

Ant colony algorithms have several interesting characteristics; we shall mention in particular high *intrinsic parallelism*, *flexibility* (a colony of ants is able to adapt to modifications of the environment), *robustness* (a colony is able to maintain its activity even if some individuals fail), *decentralization* (a colony does not obey a centralized authority), and *self-organization* (a colony finds a solution, which is not known in advance, by itself). This method seems particularly useful for problems which are *distributed* in nature, problems of dynamic *evolution*, and problems which require *strong fault tolerance*. At this stage of development of these recently introduced algorithms, however, their application to particular optimization problems is not trivial: it must be the subject of a specific treatment, which can be difficult.

1.3.5 Other Metaheuristics

Whether other metaheuristics are variants of the most famous methods or not, they are legion. The interested reader can refer to Chaps. 9 and 10 of this book and three other recent books [15, 17, 19] each one of which is devoted to several metaheuristics.

1.4 Extensions of Metaheuristics

We review here some of the extensions which have been proposed to deal with some special features of optimization problems.

1.4.1 Adaptation for Problems with Continuous Variables

Problems with continuous variables, by far the most which are common ones in engineering, have attracted less interest from specialists in informatics. The majority of metaheuristics, which are of combinatorial origin, can however be adapted to the continuous case; this requires a discretization strategy for the variables. The discretization step must be adapted in the course of optimization to guarantee at the same time the regularity of the progression towards the optimum and the precision of the result. Our proposals relating to simulated annealing, the tabu method, and GAs are described in [3, 4, 21].

1.4.2 Multiobjective Optimization

More and more problems require the simultaneous consideration of several contradictory objectives. There does not exist, in this case, a single optimum; instead, one seeks a range of solutions that are "Pareto optimal," which form the "compromise surface" for the problem considered. These solutions can be subjected to final arbitration by the user. The principal methods of multiobjective optimization (either using a metaheuristic or not) and some applications, in particular in telecommunications, were presented in the book [6].

1.4.3 Hybrid Methods

The rapid success of metaheuristics is due to the difficulties encountered by traditional optimization methods in complex engineering problems. After the initial success of using various metaheuristics, the time came to make a realistic assessment and to accept the complementary nature of these new methods, both with other methods of this type and with other approaches: from this, we saw the current emergence of *hybrid methods* (see for example [18]).

1.4.4 Multimodal Optimization

The purpose of multimodal optimization is to determine a whole set of optimal solutions, instead of a single optimum. Evolutionary algorithms are particularly well adapted to this task, owing to their distributed nature. The variants of the "multi-population" type exploit several populations in parallel, which endeavor to locate different optima.

1.4.5 Parallelization

Multiple modes of parallelization have been proposed for the various metaheuristics. Certain techniques were desired to be general; others, on the other hand, benefit from specific characteristics of the problem. Thus, in problems of placement of components, the tasks can be naturally distributed between several processors: each one of them is responsible for optimizing a given geographical area and information is exchanged periodically between nearby processors (see, for example, [20, 22]).

1.5 Place of Metaheuristics in a Classification of Optimization Methods

In order to recapitulate the preceding considerations, we present in Fig. 1.11 a general classification of mono-objective optimization methods, already published in [6]. In this figure, one can see the principal distinctions made above:

- Initially, combinatorial and continuous optimizations are differentiated.
- For combinatorial optimization, one can approach the problem by several different methods when one is confronted with a hard problem; in this case, a choice is sometimes possible between "specialized" heuristics, entirely dedicated to the problem considered, and a metaheuristic.
- For continuous optimization, we immediately separate the linear case (which is concerned in particular with *linear programming*) from the nonlinear case, where the framework for hard optimization can be found. In this case, a pragmatic solution can be to resort to the repeated application of a local method which may or may not exploit the gradients of the objective function. If the number of local minima is very high, recourse to a global method is essential: those metaheuristics are then found which offer an alternative to the traditional methods of global optimization, those requiring restrictive mathematical properties of the objective function.
- Among the metaheuristics, one can differentiate the metaheuristics of "neighborhood," which make progress by considering only one solution at a time (simulated annealing, tabu search, …), from the "distributed" metaheuristics, which handle in parallel a complete population of solutions (genetic algorithms, …).

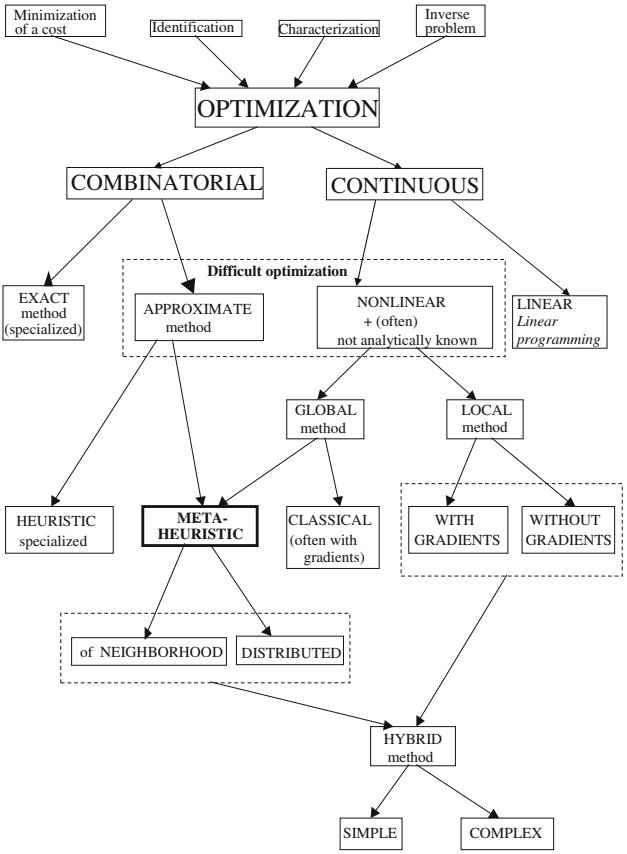


Fig. 1.11 General classification of mono-objective optimization methods

- Finally, hybrid methods often associate a metaheuristic with a local method. This cooperation can take the simple form of relaying between the metaheuristic and the local technique, with the objective of refining the solution. But the two approaches can also be intermingled in a more complex way.

1.6 Applications of Metaheuristics

Metaheuristics are now regularly employed in all sectors of engineering, to such an extent that it is not possible to draw up a complete inventory of the applications here. Several examples will be described in the chapters devoted to various metaheuristics. The last part of this book is devoted to a detailed presentation of three case studies, in the fields of logistics systems, air traffic, and vehicle routing.

1.7 An Open Question: The Choice of a Metaheuristic

This presentation must not ignore the principal difficulty with which an engineer is confronted in the presence of a concrete optimization problem: that of the choice of an "efficient" method, able to produce an "optimal" solution—or one of acceptable quality—in "reasonable" computing time. In relation to this pragmatic concern of the user, the theory is not yet of great help, because convergence theorems are often nonexistent or applicable only under very restrictive assumptions. Moreover, the "optimal" adjustment of the various parameters of a metaheuristic that might be recommended theoretically is often inapplicable in practice, because it induces a prohibitive computing cost. Consequently, the choice of a "good" method, and the adjustment of the parameters of that method, generally calls upon the know-how and "experience" of the user, rather than the faithful application of well-laid-down rules. The research efforts in progress, for example the analysis of the "energy landscape" or the development of a taxonomy of hybrid methods, are aimed at rectifying this situation, which is perilous in the long term for the credibility of metaheuristics. Nevertheless, we will try to outline, in Chap. 13 of this book, a technique that may be of assistance in the selection of a metaheuristic.

1.8 Outline of the Book

This book comprises three parts.

The first part is devoted to a detailed presentation of the more widely known metaheuristics:

- the simulated annealing method (Chap. 2);
- tabu search (Chap. 3);
- variable neighborhood search (Chap. 4);
- the GRASP method (Chap. 5);
- evolutionary algorithms (Chap. 6);
- ant colony algorithms (Chap. 7);
- particle swarm optimization (Chap. 8).

Each one of these metaheuristics is actually a family of methods, the essential elements of which we try to discuss.

In the second part (Chaps. 9–13) we present some other metaheuristics, which are less widespread or still emergent. Then we describe some extensions of metaheuristics (constrained optimization, multiobjective optimization, …) and some ways of searching.

Lastly, we consider the problem of the choice of a metaheuristic, and we describe two unifying methods which may help to reduce the difficulty of this choice.

The last part concentrates on three case studies:

- hybrid metaheuristics designed for optimization of logistics systems (Chap. 14);
- metaheuristics aimed at solving vehicle routing problems (Chap. 15);
- applications in air traffic management (Chap. 16).

References

1. Berthiau, G., Siarry, P.: État de l'art des méthodes d'optimisation globale. RAIRO Operations Research **35**(3), 329–365 (2001)
2. Cerny, V.: Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. Journal of Optimization Theory and Applications **45**(1), 41–51 (1985)
3. Chelouah, R., Siarry, P.: A continuous genetic algorithm designed for the global optimization of multimodal functions. Journal of Heuristics **6**, 191–213 (2000)
4. Chelouah, R., Siarry, P.: Tabu Search applied to global optimization. European Journal of Operational Research **123**, 256–270 (2000)
5. Cohoon, J., Hegde, S., Martin, W., Richards, D.: Distributed genetic algorithms for the floorplan design problem. IEEE Transactions on Computer-Aided Design **10**(4), 483–492 (1991)
6. Collette, Y., Siarry, P.: *Multiobjective Optimization*. Springer (2003)
7. Colorni, A., Dorigo, M., Maniezzo, V.: Distributed optimization by ant colonies. In: *Proceedings of the European Conference on Artificial Life, ECAL'91*, pp. 134–142. Elsevier (1991)
8. Fogel, L.J., Owens, A.J., Walsh, M.J.: *Artifical Intelligence Through Simulated Evolution*. Wiley (1966)
9. Fraser, A.S.: Simulation of genetic systems by automatic digital computers. Australian Journal of Biological Sciences **10**, 484–491 (1957)
10. Glover, F.: Future paths for integer programming and links to artificial intelligence. Computers and Operations Research **13**(5), 533–549 (1986)
11. Glover, F., Laguna, M.: *Tabu Search*. Kluwer Academic (1997)
12. Goldberg, D.E.: *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley (1989)
13. Holland, J.H.: Outline for logical theory of adaptive systems. Journal of the Association for Computing Machinery **3**, 297–314 (1962)
14. Kirkpatrick, S., Gelatt, C., Vecchi, M.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983)
15. Pham, D., Karaboga, D.: *Intelligent Optimisation Techniques*. Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks. Springer (2000)
16. Rechenberg, I.: *Cybernetic Solution Path of an Experimental Problem*. Royal Aircraft Establishment Library Translation (1965)
17. Reeves, C.: *Modern Heuristic Techniques for Combinatorial Problems*. Advanced Topics in Computer Science Series. McGraw-Hill Ryerson (1995)
18. Renders, J., Flasse, S.: Hybrid methods using genetic algorithms for global optimization. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics **26**(2), 243–258 (1996)
19. Saït, S., Youssef, H.: *Iterative Computer Algorithms with Applications in Engineering*. IEEE Computer Society Press (1999)
20. Sechen, C.: *VLSI Placement and Global Routing Using Simulated Annealing*. Kluwer Academic (1988)
21. Siarry, P., Berthiau, G., Durbin, F., Haussy, J.: Enhanced Simulated Annealing for globally minimizing functions of many continuous variables. ACM Transactions on Mathematical Software **238**, 209–228 (1997)
22. Wong, D., Leong, H., Liu, C.: *Simulated Annealing for VLSI Design*. Kluwer Academic (1988)

http://www.springer.com/978-3-319-45401-6

Metaheuristics
Siarry, P. (Ed.)
2016, XVI, 489 p. 181 illus., 69 illus. in color., Hardcover
ISBN: 978-3-319-45401-6