CONTROL, SYSTEMS
AND INDUSTRIAL ENGINEERING SERIES

Optimization in Engineering Sciences

Metaheuristics, Stochastic Methods and Decision Support

Dan Stefanoiu, Pierre Borne
Dumitru Popescu, Florin Gh. Filip
and Abdelkader El Kamel

iSTE WILEY

Dan STEFANOIU
Pierre BORNE
Dumitru POPESCU
Florin Gheorghe FILIP
Abdelkader EL KAMEL

Optimization in Engineering Sciences

Metaheuristics, Stochastic Methods and Decision Support

Wiley 2014

This book was realized with the support of European Project FP7 ERRIC (*Empowering Romanian Research on Intelligent Information Technology*), contract no. FP7-REGPOT-2010-1/264207.

Table of Contents

List of figures

List of tables

List of algorithms

Acronyms

ACA	Ant colony algorithm(s)
AHP	Analytic hierarchy process
AI	Artificial intelligence
ANP	Analytic network process
AP	Achieved performance(s)
APSOA	Adaptive particle swarm optimization algorithm
AR	Auto-regressive (model(s), class, component, etc.)
ARMA	Auto-regressive model (class) with moving average
ARMAX	Auto-regressive model (class) with moving average and exogenous control
BA	Baker's (genuine) algorithm
BatA	Bats algorithm(s)
BeeA	Bees algorithm(s)
BGA	Baker's generalized algorithm
BPRS	Binary pseudo-random sequence
DM	Decision model
DNA	Deoxyribonucleic acid
DSS	Decision Support Systems
EV	Expected value (statistical mean)
FA	Fireflies algorithm(s)
GA	Genetic algorithm(s)
GRASP	Greedy randomized adaptive search procedure(s)
GT	Games Theory
HITA	"How is this (objective) attained?" (decision test)
IPH	Implicit parallelism hypothesis (of Holland)
HSA	Harmony search algorithm(s)
IT	Information technology
LSB	Least significant bit
LSM	Least squares method
MA	Moving average (model(s), class, component, etc.)
MAP	Multi-attribute problem(s)
MAUT	Multi-attribute utility theory
MCDA	Multi-criteria decision analysis
MCDM	Multi-criteria decision making
MCP	Multi-criteria problem(s)
MOP	Multi-objective problem(s)
MSB	Most significant bit
N-PRS	Pseudo-random sequence with normal (Gaussian) distribution

N-PRSG	Normal (Gaussian) pseudo-random sequences generator(s)
NC	Nominal controller
NM	Nominal model
NS	Nominal system
NP	Nominal performance(s)
OWA	Ordered weights averaging method
P-PRSG	Prescribed probability distribution pseudo-random sequences generator(s)
PQ	Prediction quality
PRS	Pseudo-random sequence (of numbers)
PRSG	Pseudo-random sequence generator(s)
PSO	Particle swarm optimization
PSOA	Particle swarm optimization algorithm(s)
RS	Real system
SACA	Systemic ant colony algorithm(s)
SI	System identification
SNR	Signal-to-noise ratio
SPSOA	Standard particle swarm optimization algorithm
ST	System theory
SUS	Stochastic Universal Sampling
s.t.	subject to (restrictions or constraints)
TOPSIS	Technique for ordering by similarity to ideal solution
U-PRS	Uniformly distributed pseudo-random sequence of numbers
U-PRSG	Uniformly distributed pseudo-random sequences generator
WDTM	"What does this mean?" (decision test)
WITI	"Why is this (objective) important?" (decision test)

Preface

This book is the result of collaboration between teachers and researchers from France and Romania, with the support of FP7 ERRIC European project. The goal was to complete the book of *Optimization in Engineering Sciences*, which has been published in 2013 by John Wiley & Sons and ISTE, as well. The first volume was concerned with the presentation of *exact optimization methods* at the service of engineers and decision-makers.

In case of uncertain or poorly defined problems, possibly subject to random perturbations or for which the search for solutions might evolve towards the combinatorial explosion, the exact methods are very unlikely to provide solutions within acceptable period of time.

The methods presented in this volume allow finding, if not the optimal solution of a problem, at least a good solution in acceptable run times.

The methods presented here are:

- Metaheuristics: local approaches (such as the simulated annealing, the Tabu search, etc.) and global approaches (such as the evolutionary algorithms, ant colonies, particle swarms, etc.). These methods are mainly based on nature.
- The stochastic approaches, taking into account the uncertainties of various data sources and including some aspects of the game theory.
- Multi-objective optimization.
- The methods and techniques for decision-making, also including some aspects of the game theory and for which numerous references to existing software are pointed out.
- The use of simulation for decision-making.

Throughout the book, various examples are presented, in order to illustrate the proposed methods, while the possible application fields of the concerned algorithms are indicated.

The authors
Bucharest and Lille
July 2014

Acknowledgements

The authors are very grateful to Dr. L. Popescu, Dr. J. Culita, Dr. F. Popa and Miss D. Gherghina, who kindly accepted to be involved in translating this book from French to English and/or to perform professional English proof of the whole manuscript.

Chapter 1

Metaheuristics – Local Methods

1.1. Overview

The term of *heuristic* optimization originates from ancient Greek, where *heuriskein* (*χευρισκειν*) means "to discover", "to learn". There is a more subtle meaning of this word tough, as revealed by the following example. Assume the reader of this book is challenged to find and mark in text the position of the words *metaheuristic* or *metaheuristics*. Each of us has a specific strategy to answer the challenge. Nevertheless, in the whole, two categories of readers exist: the reader who systematically analyses the text, trying not to miss occurrences and the reader who approaches the text "diagonally", relying on his/her visual capacity of pattern recognition, without actually looking at every word. One says that the first reader performs an *exhaustive search* (like a computer), whilst the second reader makes a *heuristic search* (like a living entity, not necessarily consciously). Obviously, the research duration of the first reader usually is longer than of the second reader. However, it is very likely that the first reader will be able to find the most occurrences, whilst the second one could miss enough occurrences. Finally, when comparing the performance of the two readers, the number of occurrences found by the first one is surprisingly close to the number of occurrences marked by the second one, despite the big difference between the search durations.

The first two chapters of this book are devoted to the description of a collection of optimization methods that simulate the second type of readers' attempt. Such methods actually are inspired either by the behavior of biological entities/systems or by the evolution of some natural phenomena. The discussion is focused next on a special class of optimization problems in engineering, more specifically on the class of *granular* optimization. This concept is explained in the following.

The optimization problem of concern is formulated in context of a cost function (or criterion), as defined below:

$$f : \mathcal{S} \to \mathbb{R}, \qquad [1.1]$$

where *the search space* $\mathcal{S}$ usually is a bounded subset of $\mathbb{R}^{nx}$. (Sometimes, f is referred to as *fitness*.) What makes the criterion f so special? On one hand, it rather has a fractal variation, with many ruptures (and thus with many local extremes). On the other hand, it is quite difficult (if not impossible) to evaluate its derivatives in complete form (provided they exist). In terms of regularity, the f criterion generally is locally continuous or derivable, but this property does not extend to the global variation. (The criterion could globally be smooth, but this usually happens very seldom.) In turn, one can evaluate the f criterion for each point $\mathbf{x}$ of research space $\mathcal{S}$, according to some a priori known mathematical expression. Thus, in order to solve the optimization problem:

$$\underset{\mathbf{x}\in\mathcal{S}}{\text{opt}}\ f, \qquad [1.2]$$

which means to find the global optimum of f and the optimum point $\mathbf{x}^{\text{opt}} \in \mathcal{S}$, it only is possible to compare various criterion values in points of research space. As the search has to end after a reasonable duration, only a finite discrete subset of $\mathcal{S}$, say $\mathcal{D}$, can be used in this aim. One seeks not to miss the global optimum and therefore the $\mathcal{D}$ subset has to include a very large number of points to test.

The problem [1.2] is then relaxed, being replaced by the following problem:

$$\underset{\mathbf{x}\in\mathcal{D}\subset\mathcal{S}}{\text{opt}}\ f, \qquad [1.3]$$

where, as already mentioned, $\mathcal{D}$ is a finite discrete subset. Due to their large number, the test points of $\mathcal{D}$ are referred to as *granules* or *grains*. Consequently, [1.3] becomes a *granular optimization problem*. Solving the problem [1.3] means in this context finding the grain of $\mathcal{D}$, which is located as close as possible to the global optimum point of f.

The following example can illustrate the principle of granulation. Assume that the following adage: « *Εν αρχεοσ ο αρητμοσ.* » (/*En arheos o aritmos.*/) has to be translated. (It belongs to the famous mathematician and physicist Archimedes.) One wants the closest translation. Obviously, the criterion here is the map between the ancient Greek and a modern language, say English. The search space $\mathcal{S}$ is a dictionary with many words (few tens of thousands), granular by nature (a grain being associated here to a word). In order to perform the translation, one may want first to insulate the "sub-dictionary" $\mathcal{D}$ including all words of $\mathcal{S}$ that begin with

α (/a/), ε (/e/) and o (/o/). Next, an appropriate search strategy has to be set, as checking all words of $\mathcal{D}$ (still very large) is unthinkable. By adopting a heuristic like strategy, the sub-dictionary size can be reduced, as soon as the next letters composing the words to translate are taken into account. Finally, $\mathcal{D}$ only includes the words to translate and one obtains thus a first but coarse translation: *in, ancient/antique, is/was, number*. A refinement is necessary though, so that the good meaning of the adage is found. A second optimization problem can thus be stated, but, this time, by accounting all synonyms of already found words. One even can add all usual expressions containing those words and synonyms. Since the size of restricted sub-dictionary stays small, one can adopt now exhaustive search as suitable strategy. One reaches then for the closest translation to the adage spirit: *The number was in the beginning*.

The methods to solve granular optimization problems should lead to numerical computer algorithms, otherwise they are not really useful in engineering. The strategies adopted in the previous example can easily be followed by a human being, but the computer needs programming in order to perform similar reasoning. Thus, first of all, the words of $\mathcal{S}$ dictionary are recorded to memory locations addressed by the ASCII codes of their letters ($\mathcal{S}$ becomes thus an *electronic dictionary*). In this manner, the exhaustive search is avoided (no need to parse all dictionary addresses until the word is found). Second, an expert system of usual expression in the modern language has to be designed and implemented. Here, again, the memory addresses have to be generated in a way that the exhaustive search can be avoided (if possible). Third, in order to increase the search speed, some statistic database pointing to the most employed words of dictionary can be built into the dictionary. The modern automatic translators are based on the principles of heuristic strategies, as stated above.

Concerning the problem [1.3], the main goal is to design solving methods that can avoid the exhaustive search or, at least, that are only adopting this strategy as a final stage, on a restricted search subset. Moreover, such methods should lead to the numerical algorithms to implement on computer. Obviously, by avoiding the exhaustive search, the global optimum could be missed, but, in turn, there is hope that the search speed increases sensibly, while the accuracy stays good enough. There is a preference for methods allowing the user to control the trade-off between the search speed and the optimal solution accuracy, in spite of their higher complexity. (Some of such methods are described within this chapter.) For the other (usually less complex) methods, it is up to the user to select or not one of them in applications.

The heuristic methods that can be implemented on a computer are referred to as *metaheuristics*. They rely on the following basic principle: the search for optimum actually is simulating either the behavior of a biologic system or the evolution of a natural phenomenon, including an intrinsic optimality mechanism. For this reason, a new Optimizations branch has developed in the past 20 years, namely the

Evolutionary Programming. All numerical algorithms designed from metaheuristics are included into this class of optimization techniques.

In general, all metaheuristics are using a pseudo-random engine to select some parameters or operations that yield estimation of optimal solution. Therefore, the procedures to generate *pseudo-random (numerical) sequences* (PRS) are crucial in metaheuristics design. When speaking about pseudo-random numbers, their probability density should a priori be specified. In case of metaheuristics, two types of probability densities generally are considered: uniform and normal (Gaussian). Thus, the following types of generators are useful:

- *uniformly distributed pseudo-random sequences generator* (U-PRSG);
- *prescribed probability distribution pseudo-random sequences generators* (P-PRSG).

In Appendices A and B of this book, algorithms of both generator types are described. They are simple and effective. Sometimes (but rather seldom), more complex and sophisticated algorithms are preferred in applications.

Unfortunately, the use of pseudo-random numbers in conjunction with metaheuristics makes impossible the formulation of any general and sound result related to convergence. The only way to deal with convergence is to state some conjectures for those metaheuristics that seemingly are successful in the most applications. If the natural optimality mechanism is well simulated, there is a good chance that the corresponding metaheuristic converges towards optimum in the most cases. This is a good foundation for a realistic convergence conjecture.

Two categories of metaheuristics are described in this book: local ones et global ones. In case of local methods, one assumes that either the criterion has a unique global optimum or the search is restricted to some narrow subset including the global optimum. The goal of global methods is to find the overall optimum (or, at least, a good approximation of it), by performing a search in several zones of the S space. Usually, the search is following a scenario that allows sudden change of focusing zone (with the help of PRS), in order to avoid attraction of local optima.

1.2. Monte-Carlo principle

One of the first approaches related to granularity of numerical problems (not only of optimization kind) is known as *Monte-Carlo Method*. It was introduced by the physicist Stanislaw Ulam, in the end of the ’40s [MET 49], after trying without any success to build some realistic analytical model of « Solitaire » game (in correlation with an analytical model of matter atomic kernel). He noted that perhaps it is better to observe the game results over 1000 plays and build afterwards an approximate associated statistic model, instead of writing the equations corresponding this (extremely complicated) statistics. His idea was rapidly understood by John von Neumann, a researcher who already performed similar research in the framework of some military secret projects. The two scientists have

chosen a natural codename for this method, as inspired by gambling and casinos: *Monte-Carlo* [ECK 87]. (Actually, the name was proposed by John von Neumann, after learning about the passion for casinos of one of Sanislaw Ulam's uncles.)

The idea of this method is simple, but empirical. If a numerical description of a system or phenomenon with many entries is necessary, then it is of worth to stimulate this entity by a big number of random input values and to find a good result by following some simple computational rules. Very often, such a result is surprisingly close to the real numerical characteristic of the entity of interest. This is due to the Theorem of Big Numbers, in combination with ergodic hypotheses and the Theorem of Central Limit from Statistics. For this attempt is working correctly, the only requirement is to formulate the problem to solve in such a way that the method principle can be exploited.

The next example shows how the method works. Assume that a good approximation of Pythagoras' number π has to be found. Then, instead of using a Taylor expansion of a trigonometric map (which would constitute a sound approach), a simple geometrical property can be exploited. Since the unit square includes the unit circle and the ration between their areas is $4/\pi$, it suffices to approximate this number, without actually measuring the surfaces. According to Monte-Carlo Method, to solve this problem, one has first to fill in the square with N randomly generated points, uniformly distributed. The number of points falling into the circle, say n, are then counted. Thus, an approximate value of π can be computed as follows:

$$\pi \cong 4\frac{n}{N}. \qquad [1.4]$$

The bigger N, the more accurate the approximation. The result is illustrated by the succession of images in Figure 1.1. One starts with 3000 points. More points are added subsequently, until the desired accuracy is obtained. For 30000 points, $\pi \cong 3.1436$.

The success of method tremendously depends on the probability distribution of generated points. If non uniform, the error can rapidly grow. The entries are here the randomly generated points, which actually perform sampling of the two surfaces. It is easy to notice that the system is stimulated by infinite number of inputs and, moreover, the inputs cannot be counted (as being uniformly spread over the square).

The computing rule is based on each point position (inside or outside the circle). To automatically decide the point position, it suffices to compute its distance from the circle center and to compare it to the unit. In general, the problems to solve by Monte-Carlo Method should rely on simple enough computational rules. Otherwise, to reach for the prescribed accuracy, the procedure could be time consuming (because the computational rules apply to each stochastic entry). Even in case of example above, the computational burden becomes important when the number of generated points increases beyond some limit.

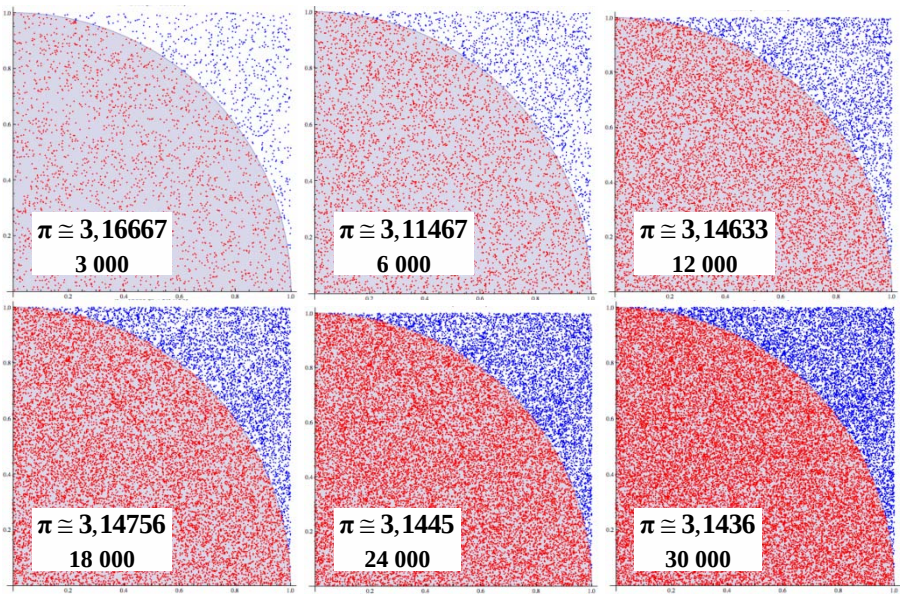


Figure 1.1. *Empirical estimation of Pythagoras' number by Monte-Carlo Method.*

As a general characteristic of Monte-Carlo Method, the procedure may be very greedy in terms of computational resources.

When looking back at the images of Figure 1.1, it is easy to see the granularity effect (the points are like grains pored on the square surface). It follows that, in context of Monte-Carlo Method, finding a good approximation of π number is a granular computational problem (not necessarily of optimization type).

The method frequently is employed in surfaces or volumes evaluation, in case of complicated geometrical shapes. More specifically, by this method, multiple integrals like below can be evaluated:

$$\int_{\mathcal{D}} f(\mathbf{x})\mathrm{d}\mathbf{x}\,, \qquad [1.5]$$

where the map f usually has simple expression, whereas the border of integration domain $\mathcal{D} \subset \mathbb{R}^{nx}$ is described by complex equations, sometimes in implicit form. It is much easier to test whether some point of $\mathbb{R}^{nx}$ space belongs or not to the integration domain instead of computing the integral [1.5]. This is the heart of Monte-Carlo principle.

The integral [1.5] can be approximated by means of grains technique:

1. The computational problem is formulated in context of $\mathbb{R}^{nx+1}$ space, where f generates a hyper-surface over the $\mathcal{D}$ domain. The integral actually is the volume of the solid body framed by $\mathcal{D}$ and the hyper-surface, say V_f.
2. The solid body can be included into a hyper-cylinder with a basis hyper-disc covering $\mathcal{D}$. For now, the hyper-cylinder height is unknown.
3. Many grains are uniformly pored on the hyper-disc.
4. One checks each grain location (inside or outside $\mathcal{D}$ domain).
5. For each grain inside $\mathcal{D}$, the value of f is computed, in order to determine the minimum and maximum heights of hyper-surface.
6. The hyper-cylinder height is set so that it completely includes the hyper-surface.
7. The hyper-cylinder volume is computed, say V_{hc}.
8. The available grains layer on hyper-cylinder basis is uniformly replicated at different heights, until the whole hyper-cylinder is filled in. All grains that can be projected on $\mathcal{D}$ domain are thus located inside the solid body, provided their height is between 0 and the (already computed) values of f.
9. All grains from the solid body are counted (by comparing their heights with f values). Denote by n this number.
10. All grains of hyper-cylinder, say N, are counted as well.
11. The volume of solid body is approximated by:

$$V_f \cong \frac{n}{N} V_{hc}. \qquad [1.6]$$

In spite of its empirical spirit, the Monte-Carlo Method is employed in many modern applications coming from physics, engineering, biology, statistics, economy, telecommunications, computer vision etc.

Nowadays, a whole family of Monte-Carlo methods exists, depending on various implementations of basic principle [FIS 95], [KRO 11]. They can be used to perform not only approximations of expressions difficult to evaluate, but also some statistical information concerning the confidence degree in such results. Moreover, modern Monte-Carlo methods are often working with non uniform probability distributions, in order to alleviate the computational burden.

For the Optimizations field, the use of Monte-Carlo methods rather is considered a rudimentary approach. In the previous example, one has noticed that, during the integral evaluation, the extremes of f were quite empirically determined. Indeed, the more grains are pored on the $\mathcal{D}$ domain, the most accurate the approximations of f extremes. However, this attempt is not a real search for optimum, in the spirit of optimization procedures, where a direction towards optimum usually is estimated and upgraded, if necessary. The search here is "blind", without considering the

properties of criterion to optimize. Moreover, in case of a real optimization problem, the criterion can be expressed by complex enough equations, which involves the grains evaluation could be very slow.

The metaheuristics have adopted the stochastic granularity principle though, as it could help to explore the whole search space for the global optimum.

To conclude this section, the Algorithm 1.1 introduces the Monte-Carlo basic procedure for optimization purposes.

Algorithm 1.1. *Monte-Carlo basic procedure for local heuristic optimization.*

1. Input data:
- Search vicinity $\mathcal{V}$ (equations allowing the user to decide whether a point belongs or not to this set).
- Optimization criterion, f .
- Minimum number of stochastic points to generate during the search, K .
- Accuracy threshold, $\varepsilon > 0$.

2. Initialization.

a. Select at random (but uniformly distributed) the starting point $\mathbf{x}_0 \in \mathcal{V}$. A U-PRSG of nx size has to be used in this aim (see section B.4 from Appendix B). If the starting point does not belong to $\mathcal{V}$, then it will be generated until this property is verified. (It is necessary to take into account the topology of $\mathcal{V}$.)

b. Evaluate the starting point performance: $f(\mathbf{x}_0)$.

c. Set the initial optimal point: $\mathbf{x}^{\text{opt}} = \mathbf{x}_0$

d. Set the starting iterations number: $k = 0$.

3. For $k \in \overline{0, K-1}$:

3.1. Generate $\mathbf{x}_k^{k+1}$ inside the vicinity $\mathcal{V}$, by means of the U-PRSG, after being well calibrated in this aim.

3.2. If $\mathbf{x}_k^{k+1} \in \{\mathbf{x}_i\}_{i \in \overline{0,k}}$, the point already exists and it must be replaced by a different one. Repeat the previous step.

3.3. Otherwise, store $\mathbf{x}_{k+1} = \mathbf{x}_k^{k+1}$ in memory.

3.4. Evaluate the performance of newly generated point: $f(\mathbf{x}_{k+1})$.

3.5. If $f(\mathbf{x}_{k+1})$ is better than $f(\mathbf{x}^{\text{opt}})$, update the optimal point: $\mathbf{x}^{\text{opt}} = \mathbf{x}_{k+1}$.

3.6. Proceed with the next iteration: $k \leftarrow k+1$.

4. For $k \geq K$:

4.1. Generate a point $\mathbf{x}_k^{k+1}$ inside the vicinity $\mathcal{V}$, by using the U-PRSG.

4.2. If $\mathbf{x}_k^{k+1} \in \{\mathbf{x}_i\}_{i\in\overline{0,k}}$, the point already exists and it must be replaced by a different one. Repeat the previous step.

4.3. Otherwise, evaluate the performance of the new point: $f\left(\mathbf{x}_k^{k+1}\right)$.

4.4. If $\left|f\left(\mathbf{x}_k^{k+1}\right) - f\left(\mathbf{x}^{\text{opt}}\right)\right| < \varepsilon$, stop the search, as no real improvement is obtained. If, moreover, $f\left(\mathbf{x}_k^{k+1}\right)$ is better than $f\left(\mathbf{x}^{\text{opt}}\right)$, then update the optimal point: $\mathbf{x}^{\text{opt}} = \mathbf{x}_k^{k+1}$. Go directly to the final stage.

4.5. Otherwise, store $\mathbf{x}_{k+1} = \mathbf{x}_k^{k+1}$ in memory.

4.6. If $f\left(\mathbf{x}_{k+1}\right)$ is better than $f\left(\mathbf{x}^{\text{opt}}\right)$, update the optimal point: $\mathbf{x}^{\text{opt}} = \mathbf{x}_{k+1}$.

4.7. Proceed with the next iteration: $k \leftarrow k+1$.

5. Return:

- The optimal point: $\mathbf{x}^{\text{opt}}$.
- The optimal performance: $f\left(\mathbf{x}^{\text{opt}}\right)$.

Usually, the numerical procedure of Algorithm 1.1 is time consuming and greedy. Starting from certain iteration, the search for duplicates among the already generated points can become much slower than the evaluation of performance for the newly generated point. However, this step is absolutely necessary in the 3-rd stage of the algorithm, in order to ensure the minimum degree of granularity and to avoid inconsistent results. On the contrary, this step could miss in the 4-th stage, especially when the performance is evaluated faster than the search for duplicates. (Thus, the step 4.2 can be removed from the procedure, in order to speed up the search.)

This procedure is integrated into some of the metaheuristics from this chapter.

1.3. Hill climbing

One starts with a metaheuristic of great simplicity, but that reveals, on one hand, the basic principle of PRS generators use (that allow advancing towards the optimum) and, on the other hand, a link to the *Artificial Intelligence* (AI) [RUS 95]. Like this method, many other metaheuristics are (directly or indirectly) bond to AI techniques. Moreover, this method is an example of how the Monte-Carlo principle can be exploited in heuristic optimization.

From the beginning, one assumes that the f criterion has to be maximized in a vicinity $\mathcal{V}$ inside $\mathcal{S}$. Since $\mathcal{S}$ est bounded, $\mathcal{V}$ inherits the same property. Thus, bounds are well known along each axis of the Cartesian space $\mathbb{R}^{nx}$. It is not mandatory however that the vicinity has the shape of a hyper-cube or that the criterion has smooth variation. Even though the criterion could exhibit more local extremes, the vicinity should be selected to embrace the global maximum.

Let $\mathbf{x}_0 \in \mathcal{V}$ be the initial point to start the search. This point could be seen as a place from which some tourist starts a journey to the top of a hill or mountain. Presumably, the tourist is not very well informed on the path to follow, so that he/she has to advance quite blindly. His/Her strategy is simple, though. Each time a higher position is conquered (comparing to the previous position), as decided by the "altitude" criterion f, the tourist tries to conserve it and to avoid to go beneath. Nevertheless, the tourist cannot leave the peak zone (i.e. the $\mathcal{V}$ vicinity). When located in current position $\mathbf{x}_k \in \mathcal{V}$, the tourist seeks to find the path to the next position, by selecting a direction at random. More specifically, the next position is evaluated as follows:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta\mathbf{x}_{k+1}, \quad \forall k \in \mathbb{N}, \tag{1.7}$$

where $\Delta\mathbf{x}_{k+1}$ is a randomly selected offset, in Monte-Carlo spirit, such that:

$$\mathbf{x}_{k+1} \in \mathcal{V} \quad \text{and} \quad f(\mathbf{x}_{k+1}) > f(\mathbf{x}_k). \tag{1.8}$$

The tourist stops when one or several conditions below are met:

- he/she cannot move, after a number of attempts, say $N \in \mathbb{N}^*$, to find the good path;
- the altitude difference between two successive positions remains too small, lower than some threshold, say $\delta > 0$, after a number of successive attempts, say $M \in \mathbb{N}^*$ (practically, the tourist cannot leave the same level contour line – the "isolevel" curve);
- the estimated gradient of his/her path, namely the vector:

$$\mathbf{f}_{k+1} = \left[\frac{f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k)}{x_{k+1,1} - x_{k,1}} \quad \frac{f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k)}{x_{k+1,2} - x_{k,2}} \quad \cdots \quad \frac{f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k)}{x_{k+1,nx} - x_{k,nx}} \right]^T, \tag{1.9}$$

  had too small norm (below some threshold $\varepsilon > 0$), during the last $M \in \mathbb{N}^*$ attempts.

The last two conditions practically are equivalent from tourist dynamics point of view, but the gradient could be almost null on peak.

The basic procedure of hill (mountain) climbing is described within Algorithm 1.2 below.

Algorithm 1.2. *Basic hill climbing procedure.*

1. Input data:
- Search vicinity $\mathcal{V}$ (equations allowing the user to decide whether a point belongs or not to this set).
- Altitude indicator, f (criterion to maximize).
- Maximum number of attempts to escape from current position, N.
- Maximum number of attempts to find a better position than the current one, M.
- Threshold to detect the isolevel curve, $\delta > 0$.

2. Initialization.

a. Select at random (but uniformly distributed) the tourist departure point $\mathbf{x}_0 \in \mathcal{V}$. A U-PRSG of nx size has to be used in this aim. If the departure point does not belong to $\mathcal{V}$, then it will be generated until this property is verified.

b. Evaluate the altitude of departure point: $f(\mathbf{x}_0)$.

c. Set a counter for the number of attempts to escape from the current position: $n = 0$.

d. Set a counter for remaining on the isolevel curve: $m = 0$.

e. Set the starting iterations number: $k = 0$.

3. Approaching the peak. For $k \geq 0$:

3.1. Perform attempts to escape from the current position:

3.1.1. Use the U-PRSG to generate an offset along some search direction: $\Delta\mathbf{x}_k^{k+1}$. The generator has to operate inside a hyper-cube that includes the vicinity, but stays as narrow as possible.

3.1.2. While $\mathbf{x}_k + \Delta\mathbf{x}_k^{k+1} \notin \mathcal{V}$, calibrate the U-PRSG to generate a new offset inside the hyper-cube $\left[\mathbf{0}, \Delta\mathbf{x}_k^{k+1}\right]$, where $\Delta\mathbf{x}_k^{k+1}$ is the most recently generated offset.

3.1.3. Set the offset: $\Delta\mathbf{x}_{k+1} = \Delta\mathbf{x}_k^{k+1}$. (Now, $\mathbf{x}_k + \Delta\mathbf{x}_{k+1} \in \mathcal{V}$ for sure.)

3.2. Test if the altitude increases if the tourist would move to the new position:

3.2.1. Evaluate the altitude of the proposed position: $f(\mathbf{x}_k + \Delta\mathbf{x}_{k+1})$.

3.2.2. If $f(\mathbf{x}_k + \Delta\mathbf{x}_{k+1}) > f(\mathbf{x}_k)$, then:

3.2.2.1. Allow the tourist to move into the new position: $\mathbf{x}_{n+1} = \mathbf{x}_n + \Delta\mathbf{x}_{n+1}$. (The altitude is already known: $f(\mathbf{x}_{k+1}) = f(\mathbf{x}_k + \Delta\mathbf{x}_{k+1})$.)

3.2.2.2. Reset the counter related to blocking on the same position: $n \leftarrow 0$.

3.2.2.3. If $f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k) < \delta$, then the tourist cannot leave the isolevel curve and, thus, the corresponding counter increases : $m \leftarrow m+1$.

3.2.2.4. Otherwise, reset the corresponding counter: $m \leftarrow 0$.

3.2.3. Otherwise, the tourist has to conserve the current position ($\mathbf{x}_{k+1} = \mathbf{x}_k$) and the blocking counter increases: $n \leftarrow n+1$.

3.3. If $n > N$ or $m > M$ stop the journey towards the peak and go to the final stage.

3.4. Otherwise, proceed with the next iteration: $k \leftarrow k+1$.

4. Return:

- The tourist current position: $\mathbf{x}_{k+1}$.
- The tourist current altitude, $f(\mathbf{x}_{k+1})$, assumed to approximate the hill peak height.

The Algorithm 1.2 was presented from informatics perspective. Therefore, it could slightly be different from other hill climbing algorithms found in the AI literature. The configuring parameters (N, M, δ) normally have implicit values, in order to help the user making a choice. Usually, the user is more likely tempted to search his/her next good path starting from the current position than to advance towards the peak in small steps. Consequently, $N \in \overline{10,100}$, while $M \in \overline{5,10}$. Concerning the δ threshold, it actually is the least significant part of altitude numerical representation. Normally, this parameter should be set at the scale of altitude indicator f. Usually, if f does not change any more, up to the last 3-7 digits, then the search can be stopped. For example, if f varies in range 0-1000, then one can set $\delta = 10^{-3}$. But, if f is 1000 times larger, maybe a better choice is $\delta = 1$.

In general, the performance of Algorithm 1.2 is modest, both in terms of accuracy and convergence speed. One first critical step is 1.a (selection of departure point). If the $\mathcal{V}$ vicinity is defined by complex equations, then the algorithm can spend long time to solve this step. It is suitable to carefully analyze both the vicinity and the criterion before configuring the algorithm. Another critical step is 3.2. This time, there is the risk to enter a very slow loop before returning inside the $\mathcal{V}$ vicinity. Nevertheless, the loop is not infinite, as the U-PRSG is enforced to work

with smaller and smaller hyper-cubes, which brings the newly generated point inside the vicinity, eventually. During the search, the algorithm can very easily be captured by a local optimum, if isolated from the global one by large and deep valleys. Moreover, the search procedure could start to oscillate. In the whole, the basic algorithm of hill climbing is easy to implement, but modest in performance.

An improved version of this procedure can be obtained if the tourist is supposed to have more traveling means, especially for guessing the next path to follow without testing too many possibilities. Of course, there is a characteristic to be accounted in this aim: the climbing directions to follow can be pointed by the gradient of altitude indicator, namely $\mathbf{f}_{\mathbf{x}}$. If the gradient could somehow be approximated, then the tourist should use this new information like a compass, in order to speed up the journey and, perhaps, to reach the peak with better accuracy.

A simple technique to estimate the gradient is to take into account two successive positions, such as $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$, like in definition [1.9]. By convention, every time the denominator is null, the corresponding gradient component is null as well. The gradient estimation is assigned to tourist position $\mathbf{x}_{k+1}$. If a gradient estimator is available, then the variable step Cauchy algorithm [BOR 13] can be employed to speed up the search. The tourist has now a compass to approximately set the orientation of the next direction to follow.

The new hill climbing procedure is described within the Algorithm 1.3.

Algorithm 1.3. *Improved hill climbing procedure, by using the Cauchy compass.*

1. Input data:
- Search vicinity $\mathcal{V}$ (equations allowing the user to decide whether a point belongs or not to this set).
- Altitude indicator, f (criterion to maximize).
- Maximum number of attempts to escape from current position, N.
- Maximum number of attempts to find a better position than the current one, M.
- Threshold to detect the isolevel curve, $\varepsilon > 0$.

2. Initialization.

a. Select at random (but uniformly distributed) the tourist departure point $\mathbf{x}_0 \in \mathcal{V}$. A U-PRSG of nx size has to be used in this aim. If the departure point does not belong to $\mathcal{V}$, then it will be generated until this property is verified.

b. Evaluate the altitude of departure point: $f(\mathbf{x}_0)$.

c. Set the initial gradient for the departure point, $\mathbf{f}_0 \in \mathbb{R}^{nx}$ (usually, $\mathbf{f}_0 = \mathbf{0}$).

d. Set the initial advancement step, $\alpha_0 \in \mathbb{R}$ (usually, $\alpha_0 = 1$).

e. Set a counter for the number of attempts to escape from the current position: $n = 0$.

f. Set a counter for remaining on the isolevel curve: $m = 0$.

g. Set the starting iterations number: $k = 0$.

3. Approaching the peak. For $k \geq 0$:

3.1. Perform attempts to escape from the current position:

3.1.1. Use the U-PRSG to generate an offset along some search direction: $\Delta\mathbf{x}_k^{k+1}$.

3.1.2. While $\mathbf{x}_k + \Delta\mathbf{x}_k^{k+1} \notin \mathcal{V}$, calibrate the U-PRSG to generate a new offset inside the hyper-cube $\left[\mathbf{0}, \Delta\mathbf{x}_k^{k+1}\right]$, where $\Delta\mathbf{x}_k^{k+1}$ is the most recently generated offset.

3.1.3. Evaluate the altitude of proposed position: $f\left(\mathbf{x}_k^{k+1}\right)$.

3.1.4. Estimate the gradient for the presumably next position, by using $\mathbf{x}_k$ and $\mathbf{x}_k^{k+1}$ in definition [1.9] (with $\mathbf{x}_k^{k+1}$ instead of $\mathbf{x}_{k+1}$). Thus, one obtains $\mathbf{f}_{k+1}$ (with null components corresponding to zero divide, if any), which is stored in memory.

3.1.5. Estimate the optimal advancement step: $\alpha_{k+1} = \alpha_k - \mathbf{f}_{k+1}^T \mathbf{f}_k$. (The previous gradient, $\mathbf{f}_k$, can be found in memory.)

3.1.6. While $\mathbf{x}_k + \alpha_{k+1}\mathbf{f}_{k+1} \notin \mathcal{V}$, calibrate the U-PRSG to generate a new advancement step in $\left[0, \alpha_{k+1}\right]$, where α_{k+1} is the most recently generated step (starting from the optimal step of 3.1.5.).

3.1.7. Set the final offset: $\Delta\mathbf{x}_{k+1} = \alpha_{k+1}\mathbf{f}_{k+1}$.

3.2. Test if the altitude increases if the tourist would move to the new position:

3.2.1. Evaluate the altitude of the proposed position: $f\left(\mathbf{x}_k + \Delta\mathbf{x}_{k+1}\right)$.

3.2.2. If $f\left(\mathbf{x}_k + \Delta\mathbf{x}_{k+1}\right) > f\left(\mathbf{x}_k\right)$, then:

3.2.2.1. Allow the tourist to move into the new position: $\mathbf{x}_{n+1} = \mathbf{x}_n + \Delta\mathbf{x}_{n+1}$. (The altitude is already known: $f\left(\mathbf{x}_{k+1}\right) = f\left(\mathbf{x}_k + \Delta\mathbf{x}_{k+1}\right)$.)

3.2.2.2. Reset the counter related to blocking on the same position: $n \leftarrow 0$.

3.2.2.3. If $\left\|\mathbf{f}_k\right\| < \varepsilon$, then the tourist practically keeps the same altitude and the corresponding counter increases: $m \leftarrow m + 1$.

3.2.2.4. Otherwise, reset the corresponding counter: $m \leftarrow 0$.

3.2.3. Otherwise, the tourist has to conserve the current position ($\mathbf{x}_{k+1} = \mathbf{x}_k$) and the blocking counter increases: $n \leftarrow n+1$.

3.3. If $n > N$ or $m > M$ stop the search and go to the final stage.

3.4. Otherwise, proceed with the next iteration: $k \leftarrow k+1$.

4. Return:

- The tourist current position: $\mathbf{x}_{k+1}$.
- The tourist current altitude, $f(\mathbf{x}_{k+1})$, assumed to approximate the hill peak height.

The complexity of Algorithm 1.3 slightly increased comparing to Algorithm 1.2. In order to better understand this procedure, the reader should analyze first the Algorithm 2.8 (the variable step Cauchy Algorithm) from [BOR 13]. The same configuring rules applies here as in case of the previous algorithm. Nevertheless, the M parameter can go down to null, since the estimated gradient is more sensitive to the directions leading to criterion extremes. In turn, the small gradient values are increasing the risk of oscillation. In this case, the search stops as soon as the norm of estimated gradient falls for the first time below the prescribed threshold ε. By difference from δ threshold in the previous algorithm, ε has to be calibrated depending on the altitude indicator variations and the scale of searching vicinity (see definition [1.9] again).

The Algorithm 1.3 performs better than Algorithm 1.2 in terms of accuracy and convergence speed. Thanks to the estimated gradient, the optimum can be found after a smaller number of iterations (even though this estimation is not that accurate). The tourist is not advancing now blindly as before, but with the help of a compass, which constitutes the main advantage.

The procedure can be adapted to find minima instead of maxima and rises no implementation issues. However, the overall performance of improved hill climbing algorithm is inferior to other metaheuristics.

1.4. Taboo search

1.4.1. *Principle*

The metaheuristic described in this section belongs to greedy descent local methods class. For this type of methods, the search starts from a possible solution $\mathbf{x}_i$ of $\mathcal{S}$. The strategy is then to focus on a local vicinity $\mathcal{V}(\mathbf{x}_i)$, in order to find another solution $\mathbf{x}_j$ that can improve the criterion current performance. The vicinity $\mathcal{V}(\mathbf{x}_i)$ corresponds to the set of all accessible solutions, after applying a single admissible movement, displacement or transition from $\mathbf{x}_i$. Usually, this set is a hyper-cube or a hyper-sphere including the current solution $\mathbf{x}_i$.

1.4.2. *Greedy descent algorithm*

Assume the goal is to minimize the f criterion. Then the search for a minimal point is performed over the vicinity $\mathcal{V}(\mathbf{x}_i)$ (corresponding to current solution $\mathbf{x}_i$) and a better solution $\mathbf{x}_j$ is recorded if $f(\mathbf{x}_j) < f(\mathbf{x}_i)$. In order to build the $\mathcal{V}(\mathbf{x}_i)$ vicinity, it suffices to generate some small offsets at random around the minimal point $\mathbf{x}_i$, such that the resulted subset is included in the search space $\mathcal{S}$. The variation range of U-PRS can be set according to the diameter of search space:

$$\mathcal{D}(\mathcal{S}) = \sup_{\mathbf{x},\mathbf{y}\in\mathcal{S}} \{\|\mathbf{x}-\mathbf{y}\|\}. \qquad [1.10]$$

For example, if $\mathcal{V}(\mathbf{x}_i)$ is set as a hyper-cube, then:

$$\mathcal{V}(\mathbf{x}_i) = \left[x_{i,1}-\gamma_{i,1}, x_{i,1}+\delta_{i,1}\right]\times\left[x_{i,2}-\gamma_{i,2}, x_{i,2}+\delta_{i,2}\right]\times\cdots\times\left[x_{i,nx}-\gamma_{i,nx}, x_{i,nx}+\delta_{i,nx}\right], \qquad [1.11]$$

where the bounds $\{\gamma_{i,n}\}_{n\in\overline{1,nx}}$ and $\{\delta_{i,n}\}_{n\in\overline{1,nx}}$ are numbers of some *uniformly distributed PRS* (U-PRS).

In case of spherical vicinities, the definition below can be employed:

$$\mathcal{V}(\mathbf{x}_i) = \left\{\mathbf{x}\in\mathcal{S} \;\middle|\; \|\mathbf{x}-\mathbf{x}_i\| < \rho_i\right\}, \qquad [1.12]$$

where the radius ρ_i is obtained with the help of a U-PRSG, after being calibrated by the diameter $\mathcal{D}(\mathcal{S})$.

In this framework, the greedy descent procedure is synthesized within the Algorithm 1.4 below.

Algorithm 1.4. *Greedy descent procedure.*

1. **Input data:**
   - Search space $\mathcal{S}$ (equations allowing the user to decide whether a point belongs or not to this set).
   - Depth indicator, f (criterion to minimize).
   - The U-PRSG bounds to generate vicinities. (Usually, such bounds are expressed as fractions of search space diameter.)
   - Minimum number of grains for Monte-Carlo Method, N.
   - Accuracy threshold for Monte-Carlo Method, $\varepsilon > 0$.

2. Initialization.

a. Select at random (but uniformly distributed) the starting point $\mathbf{x}_0 \in \mathcal{S}$. A U-PRSG of nx size has to be used in this aim. If the starting point does not belong to $\mathcal{S}$, then it will be generated until this property is verified. (It is necessary to take into account the topology of $\mathcal{S}$.)

b. Evaluate the depth of starting point: $f(\mathbf{x}_0)$.

c. Set the starting iterations number: $k = 0$.

3. For $k \geq 0$:

3.1. Call the U-PRSG (after calibration according to prescriptions) to generate the vicinity $\mathcal{V}(\mathbf{x}_k)$.

3.2. While $\mathcal{V}(\mathbf{x}_k) \not\subset \mathcal{S}$, generate a new vicinity by using U-PRSG, but after calibration according to the last generated vicinity.

3.3. Now, $\mathcal{V}(\mathbf{x}_k) \subset \mathcal{S}$. This allows finding the best solution of $\mathcal{V}(\mathbf{x}_k)$ vicinity, say $\mathbf{x}_k^{\text{opt}}$. The local search inside the vicinity $\mathcal{V}(\mathbf{x}_k)$ can be performed by using the Monte-Carlo Method (Algorithm 1.1), with granularity N and accuracy ε (already known).

3.4. Evaluate the depth of the best local solution: $f(\mathbf{x}_k^{\text{opt}})$.

3.5. If $f(\mathbf{x}_k^{\text{opt}}) \geq f(\mathbf{x}_k)$, stop the search and go to the final stage.

3.6. Otherwise, update the minimal point: $\mathbf{x}_{k+1} = \mathbf{x}_k^{\text{opt}}$.

3.7. Proceed with the next iteration: $k \leftarrow k+1$.

4. Return:

- The current minimal point: $\mathbf{x}_k$.
- The current minimal depth: $f(\mathbf{x}_k)$.

The local search procedure of step 3.3 in the Algorithm 1.4 (Monte-Carlo method) can be replaced by a different optimization technique, even withdrawn from the exact methods class [BOR 13], if allowed by the f criterion. In general, if possible, it is suitable to combine metaheuristics and exact methods in the same procedure, in order to increase the accuracy of final result. By means of metaheuristics, some vicinity of global optimum can be insulated, whereas with exact methods the vicinity can be exploited efficiently. Nevertheless, if the criterion is too fractal in nature, the use of exact methods should be avoided.

The approach above is easy to implement, but there is the risk to rapidly stop the search on a local minimum. A strategy to avoid this ending is to start the search from several initial points, uniformly spread on the search space. Many improvements of greedy descent procedure have been introduced in literature so far. One of them, quite interesting actually, is based on Taboo search, as described next.

1.4.3. *Taboo search method*

While keeping the principle of local search for minimizing a criterion, by this method, there is the possibility to jump out from the capturing vicinity and to explore a different zone. Hereafter, the term of *movement* stands for any modification allowing the search to focus on vicinity in the neighborhood of the current vicinity. As usual, the search starts from some initial point (solution) $\mathbf{x}_i$, in the vicinity $\mathcal{V}(\mathbf{x}_i)$, but it is permitted to relocate the exploitation around another point (solution) $\mathbf{x}_j \in \mathcal{V}(\mathbf{x}_i)$, even though the criterion degrades $f(\mathbf{x}_j) > f(\mathbf{x}_i)$. This actually is a movement towards another zone of interest. However, in order to avoid infinite search loops, once a solution is focused on, it will never be focused on again in the future. Thus, the focused solutions become untouchable, "taboo" (which gives the name of the method) [GLO 89], [GLO 90], [ENN 04], [GHE 07].

Assume the solution $\mathbf{x}_i$ has been visited at the k-th iteration. Then the last $N_{k,i}$ visited solutions are taboo (due to the performed movements). Denote by $\mathcal{T}_{k,i}$ the list of the last taboo solutions (*the taboo list*). According to the method principle, it is forbidden to perform movements leading to solutions from the taboo list $\mathcal{T}_{k,i}$.

The currently exploited vicinity is then:

$$\mathcal{V}_k(\mathbf{x}_i) = \mathcal{V}(\mathbf{x}_i) \setminus \mathcal{T}_{k,i}. \qquad [1.13]$$

Starting from the solution $\mathbf{x}_i$, a set of possible movements, say $\mathcal{M}_{k,i}$, can be built, during the k-th iteration. Let $m \in \mathcal{M}_{k,i}$ be such a movement. By convention, $\mathbf{x}_i \xrightarrow{m} \mathbf{x}_j$ stands for the transition from solution $\mathbf{x}_i$ to a new point $\mathbf{x}_j$, as result of movement m. Then:

$$\mathcal{V}_k(\mathbf{x}_i) = \left\{ \mathbf{x}_j \in \mathcal{V}(\mathbf{x}_i) \;\middle|\; \exists m \in \mathcal{M}_{k,i},\ \mathbf{x}_i \xrightarrow{m} \mathbf{x}_j \ \&\ \mathbf{x}_j \notin \mathcal{T}_{k,i} \right\}. \qquad [1.14]$$

Practically, definition [1.14] shows that it only is possible to generate new solutions in the vicinity of current solution.

Although this approach is quite interesting, there is no guarantee that the loops are completely avoided. Moreover, some possible solutions could become taboo even before being tested.

This fault can be removed by relaxing the definitions related to taboo labels. Thus, according to the aspiration principle, any movement leading to a better solution can be allowed in taboo zones, which opens the way towards possible solutions not yet taken into consideration.

One can notice that, often, it is easier to estimate the criterion variation $\Delta f = f(\mathbf{x}_j) - f(\mathbf{x}_i)$ than to accurately compute it for each iteration, which allows decreasing the computational burden.

The set of admissible movements can vary tremendously, depending on the application. For example, in case of traveling salesman problem, recall that any of the N cities has to be visited once and only once in a succession allowing minimizing the overall traveled distance. The following movement types can be considered here, in the string of visited cities:

- displacement of a city: CDEGAF**H**JKI $\xrightarrow{dis}$ CDE**H**GAFJKI ;
- permutation of two cities: CD**E**GAFH**J**KI $\xrightarrow{per}$ CD**J**HGAF**E**KI ;
- inversion of visiting succession for a cities group:

$$\text{CD}\boxed{\textbf{EGAF}}\text{HJKI} \xrightarrow{inv} \text{CD}\boxed{\textbf{FAGE}}\text{HJKI} .$$

In this example, the solution $\mathbf{x}_i$ is CDEGAFHJKI, which means the cities can be visited in a certain order. The movements are changing this order on purpose, to find another possible solutions, $\mathbf{x}_j$.

1.4.4. *Taboo list*

The definition and the size of taboo list are important parameters of search. If the list is too large, then search is restricted to small areas, but the risk to miss the global optimum becomes important. On the contrary, if the list is too small, it is very likely that the search is slowed down by a loop.

The common solution is to work with constant length taboo list. In this case, the most recently visited solution (as result of the last performed movement) enters the taboo list, whilst the oldest solution is removed from the list, as soon as the list has reached its maximum length. The taboo list thus acts like a last-in-first-out (LIFO) stack.

However, in many applications, it is more useful to work with adaptive taboo list length, say $N_{k,i}$, which can be modified at each iteration between some bounds:

$$N_{\min} \le N_{k,i} \le N_{\max} , \quad \forall k,i \in \mathbb{N} . \qquad [1.15]$$

Many adaptation rules were introduced so far in the literature. Two of them seem to be effective in real applications, as follows:

- If the currently generated solution improves the criterion, then the taboo list length is decreased by 1: $N_{k,i} \leftarrow \max\left\{N_{k,i}-1, N_{\min}\right\}$. Thus, the oldest two solutions are removed from the list, while the most recent one is entering the list. If the list length falls below the lower limit, then either the oldest solution is removed only or the length decreasing is postponed.
- If the currently generated solution does not improve the criterion, then the length is increased by 1: $N_{k,i} \leftarrow \min\left\{N_{k,i}+1, N_{\max}\right\}$. Thus, the solution enters the list and no other solution is removed. If the list length becomes too big (larger than the upper limit), then the oldest solution is removed.

Notice that the taboo labeling applied on the latest movements can block the search (as revealed in a future example). In this case, it is necessary to allow violation of taboo principle, in order to escape from the trap and jump to another possible solutions to test.

1.4.5. *Taboo search algorithm*

The taboo search procedure is described in Algorithm 1.5.

Algorithm 1.5. *Taboo search procedure.*

1. Input data:
- Search space $\mathcal{S}$ (equations allowing the user to decide whether a point belongs or not to this set).
- Optimization criterion, f.
- Types of possible movements starting from a solution (all restrictions accounted, if any).
- Minimum number of solutions for the current vicinity, N_{η}.
- Bounds of taboo list length: $N_{\min}$ and $N_{\max}$, with $N_{\min} < N_{\max}$.
- Maximum number of iterations, K.
- Accuracy threshold, $\varepsilon > 0$.
- Maximum number of found solutions that comply with the accuracy threshold, M.
- Maximum number of iterations to stop the search since the optimal solution did not change, N.

2. Initialization.

a. Select at random (but uniformly distributed) the starting point $\mathbf{x}_{0,0} \in \mathcal{S}$. A U-PRSG of nx size has to be used in this aim. If the starting point does not belong to $\mathcal{S}$, then it will be generated until this property is verified. (It is necessary to take into account the topology of $\mathcal{S}$.)

b. Evaluate the starting point performance: $f\left(\mathbf{x}_{0,0}\right)$.

c. Initialize the optimal solution: $\mathbf{x}^{\text{opt}} = \mathbf{x}_{0,0}$.

d. Initialize the taboo list: $\mathcal{T}_{0,0} = \left\{\mathbf{x}_{0,0}\right\}$.

e. Set the initial counter associated to the found solutions that comply with the accuracy threshold (the *threshold counter*): $m = 0$.

f. Set the initial counter associated to the number of iterations during which the optimal solution did not change (the *blocking counter*): $n = 0$.

g. Set the position of the initial optimal solution: $i = 0$.

h. Set the starting iterations number: $k = 0$.

3. For $k \in \overline{0, K-1}$:

3.1. Specify all possible movements (by accounting all restrictions, if any), starting from the current solution, $\mathbf{x}_{k,i}$. Although the set of such movements if finite, its size could be very large.

3.2. Initialize the vicinity of current solution: $\mathcal{V}_k\left(\mathbf{x}_{k,i}\right) = \varnothing$.

3.3. While $\mathcal{V}_k\left(\mathbf{x}_{k,i}\right)$ includes less than $N_{\mathcal{V}}$ points, do:

3.3.1. Call the U-PRSG to select a possible movement, pm. (If there are no possible movements to select, break the loop and jump to step 3.4.)

3.3.2. Perform the pm movement to get to a new point:

$$\mathbf{x}_{k,i} \xrightarrow{pm} \mathbf{x}_{k,i}^{pm}.$$

3.3.3. If the new point belongs to the search space ($\mathbf{x}_{k,i}^{pm} \in \mathcal{S}$), but does not belong to the taboo list ($\mathbf{x}_{k,i}^{pm} \notin \mathcal{T}_{k,i}$), then add the point to the vicinity:

$$\mathcal{V}_k\left(\mathbf{x}_{k,i}\right) \leftarrow \mathcal{V}_k\left(\mathbf{x}_{k,i}\right) \cup \left\{\mathbf{x}_{k,i}^{pm}\right\}.$$

3.3.4. Otherwise, keep the vicinity unchanged.

3.4. If the vicinity includes at least one point ($\mathcal{V}_k\left(\mathbf{x}_{k,i}\right) \neq \varnothing$):

3.4.1. Solve the problem below by exhaustive search:

$$\mathbf{x}_{k,j} = \underset{\mathbf{x} \in \mathcal{V}_k\left(\mathbf{x}_{k,i}\right)}{\operatorname{argopt}} f(\mathbf{x}).$$

Here, $\mathbf{x}_{k,j}$ is the new solution (the firstly found better point) and j points to its position inside the vicinity.

3.4.2. Update the new solution: $\mathbf{x}_{k+1,j} = \mathbf{x}_{k,j}$.

3.4.3. Add the new solution to the taboo list:

$$\mathcal{T}_{k+1,j} = \mathcal{T}_{k,i} \cup \left\{ \mathbf{x}_{k+1,j} \right\}$$

3.4.4. Update the solution position: $i = j$.

3.4.5. If the performance of new solution ($f\left(\mathbf{x}_{k+1,i}\right)$, already computed) is better than the current optimal performance ($f\left(\mathbf{x}^{\text{opt}}\right)$, available) :

3.4.5.1. Update the optimal solution: $\mathbf{x}^{\text{opt}} = \mathbf{x}_{k+1,i}$.

3.4.5.2. Reset the blocking counter: $n \leftarrow 0$.

3.4.5.3. If $\left| f\left(\mathbf{x}_{k+1,i}\right) - f\left(\mathbf{x}^{\text{opt}}\right) \right| < \varepsilon$, increment the threshold counter: $m \leftarrow m+1$.

3.4.5.4. If $m > M$, stop the search, as no real progress is made. Go directly to final stage (no. 4).

3.4.5.5. If possible, remove one or two old solutions from taboo list, such that the number of remaining taboo solutions is at least $N_{\min}$. (If this number already is smaller than $N_{\min}$, then keep unchanged the taboo list.)

3.4.6. Otherwise, the optimal solution does not change and then:

3.4.6.1. Increment the blocking counter: $n \leftarrow n+1$.

3.4.6.2. If $n > N$, stop the search, as very likely, the algorithm has reached the best solution that can be found with this procedure. Go directly to final stage (no. 4).

3.4.6.3. Otherwise, check whether the updated taboo list ($\mathcal{T}_{k+1,i}$) includes or not too many solutions (more than $N_{\max}$). In case the number is too big, remove the oldest solutions in order to keep only the most recent $N_{\max}$ taboo solutions.

3.4.6.4. Reset the threshold counter: $m = 0$.

3.4.7. Proceed with the next iteration: $k \leftarrow k+1$.

3.5. Otherwise, since the current vicinity is void, the procedure is blocked and the main loop has to be broken. Go directly to final stage (no. 4).

4. Return:

- The current optimal point: $\mathbf{x}_k$.
- The current optimal performance: $f\left(\mathbf{x}_k\right)$.

Like in case of previous procedures, several stop tests are integrated within the Algorithm 1.5. First, the search cannot overpass a maximum number of iterations, K . If this condition enforces the procedure to stop prematurely, it is recommended to increase K and to restart the algorithm from the current taboo list and optimal

solution of previous run. Second, the search can stop when either the last M optimal solutions of taboo list are not really improving the criterion performance or the optimal solution resisted for at least N successive iterations, without being overthrown. The last two stop tests actually are normal exits for a well designed taboo procedure.

The taboo list length has been limited, in order to speed up the search. (If the taboo list is too large, looking for duplicated solutions could be lengthy.) In turn, the risk of the algorithm to be captured into a loop or to oscillate has increased. Note however that the parameters M and N are preventing infinite loops. The procedure is exposed to the danger to be captured by a local optimum with big attraction force (i.e. quite well insulated from the global optimum). In this case, it is wise to restart several times the taboo procedure from different initializations.

The configuration parameters of Algorithm 1.5 (especially K, M, N and ε), can be set according to similar recommendations like for Algorithm 1.3. For example. $K \in \overline{100,200}$, $M \in \overline{5,10}$, $N \in \overline{10,30}$, while ε is determined by the variation range of f criterion. Of course, the taboo list length has to be adjusted according to M et especially N.

The step 3.1 of Algorithm 1.5 should be managed carefully. Come back to the traveling salesman example, in order to better understand how the possible movements can be selected. If CDEGAFHJKI is an initial solution, then recall that only 3 types of movements can be employed to find a different solution. Each one of such movements is well defined by some parameters, as follows:

- displacement can be performed if the position of the city to be displaced and the position where to displace it are known;
- permutation requires knowing the two positions of cities exchange;
- inversion needs the position of cities group and the number of cities within the group.

When varying all those parameters, a large number of possible movements is obtained. Testing all possible movements actually means to perform exhaustive search, which might not be a good strategy. Therefore, the Monte-Carlo principle could be adopted (like in step 3.3. of the algorithm). The effective implementation of this principle depends on each application (as the possible movements have to be indexed accordingly, to facilitate the use of some U-PRSG). The exhaustive search is a strategy to consider only if the number of possible movements is small enough.

Concerning the constraints, in case of traveling salesman, one can imagine for example that the journey from city A to city B must include a passage through the city C. Consequently, it is impossible to exchange the cities A and C or B and C. The constraints can change from iteration to iteration, depending on the current optimal solution. For example, different constraints apply to solutions CDEGAFHJKI and CDFAGEHJKI. If the path from A to H has to include F, then, in the first solution (CDEGAFHJKI), A cannot exchange with F. In the second solution (CDFAGEHJKI), exchanging A and F is possible, but, in this case, the

cities group GE cannot lie between A, F and H. So, it is necessary to move GE to another zone.

1.4.6. *Intensification and diversification*

In order to have a better guidance for the search process and, at the same time, to reduce its duration, two actions are possible:

- intensification, which yields more intense exploitation of the current zone;
- diversification, which allows exploration of various regions.

In general, a trade-off between the two actions has to be found, in order to propose solutions with good accuracy, after reasonable search durations.

The most commonly employed intensification approaches are the following:

- reducing the length of taboo list, as soon as the solution improves;
- selecting more often some preferred solutions, with closer performance to the best found solutions.

As of diversification, the usual approaches are listed below:

- sudden change of focus in the zones where there are solutions waiting to be visited;
- multiple re-runs, starting from various initializations, selected at random;
- exclusion of the most visited solutions;
- penalties applied to the solutions nearby the current solution.

1.4.7. *Application examples*

1.4.7.1. *Searching the smallest value on a table*

Table 1.1 exhibits a matrix of expenses rates made by a traveling professional in charge with the maintenance of a computer network. Each row of the table is associated to a working point, whereas each column shows the expenses rates after a traveler's journey passing through the working points. Somewhere in the table there is a minimum rate of expenses. The problem is then to find the table cell hosting the minimum expenses rate by using taboo search metaheuristic.

Table 1.1. *Expenses rates for computer network maintenance missions.*

7	9	8	11	10	7	9	7	8	5	9	6	7	10	3	4	3
8	6	10	9	8	6	14	17	16	18	16	14	13	8	5	2	7
6	4	5	4	7	8	10	13	10	12	14	16	13	6	3	6	8
7	3	7	3	5	4	7	11	10	19	13	16	15	13	8	9	7
9	6	6	4	8	11	13	14	6	18	17	15	13	11	7	10	9
7	7	8	7	9	7	15	13	16	12	14	16	14	15	14	17	10

Two randomly chosen initializations are marked on the table. Before starting the taboo search, all possible movements as well as the maximum length of taboo list

have to be specified. For each cell in the table, the admissible movements towards the neighbor cells are illustrated in Figure 1.2.

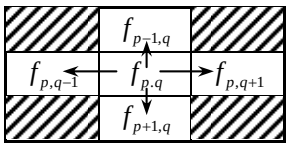


Figure 1.2. *Admissible movements in the problem of minimum expenses rates for computer network maintenance missions.*

It follows that the set of possible movements starting from the current solution $\mathbf{x}_i = (p,q)$ is the following, for each iteration $k \in \mathbb{N}$:

$$\mathcal{M}_{k,i} = \left\{ (p,q) \xrightarrow{\text{up}} (p-1,q),\ (p,q) \xrightarrow{\text{down}} (p+1,q),\ \ldots \right.$$
$$\left. (p,q) \xrightarrow{\text{left}} (p,q-1),\ (p,q) \xrightarrow{\text{right}} (p,q+1) \right\}. \qquad [1.16]$$

Since the number of movements is small, the exhaustive search is well suited (no need to implement the Monte-Carlo Method).

The taboo list can include up to 10 solutions and cannot by void (at least one solution has to receive the taboo label).

For the first initialization (on the left side in Table 1.1), the taboo search is blocked, as illustrated in Table 1.2. This actually is the effect of the test in step 3.5 from Algorithm 1.5, which enforces the procedure to end prematurely. In this case, the optimal solution is:

$$\mathbf{x}^{\text{opt}} = (4,2), \quad f\left(\mathbf{x}^{\text{opt}}\right) = f_{4,2} = 3. \qquad [1.17]$$

Table 1.2. *Taboo search leading to blocking.*

7	9	8	11	10	7	9	7	8	5	9	6	7	10	3	4	3
8	**6**	10	9	8	6	14	17	16	18	16	14	13	8	5	2	7
6	**4**	**5**	**4**	7	8	10	13	10	12	14	16	13	6	3	6	8
7	**3**	**7**	**3**	5	4	7	11	10	19	13	16	15	13	8	9	7
9	**6**	**6**	**4**	8	11	13	14	6	18	17	15	13	11	7	10	9
7	7	8	7	9	7	15	13	16	12	14	16	14	15	14	17	10

The corresponding taboo list includes 10 solutions:

$$\mathcal{T}_{9,2} = \{(2,2), (3,2), (4,2), (5,2), (5,3), (5,4), (4,4), (3,4), (3,3), (4,3)\}, \qquad [1.18]$$

while the number of iterations is 9 (count the number of arrows in Table 1.2).

The last found position in the current vicinity is $(4,3)$ (after a movement down). In fact, the last vicinity only includes two elements: $\{(2,3),(4,3)\}$ (according to Taboo Method principle).

The second initialization lead to the iterations in Table 1.3.

Table 1.3. *Taboo search leading to optimum solution.*

7	9	8	11	10	7	9	7	8	5	9	6	7	10	3	4	3
8	6	10	9	8	6	14	17	16	18	16	14	13	8	**5**	**2**	7
6	4	5	4	7	8	10	13	**10**	**12**	**14**	16	13	6	**3**	6	8
7	3	7	3	5	4	7	11	**10**	19	**13**	**16**	15	13	**8**	9	7
9	6	6	4	8	11	13	14	**6**	18	17	**15**	**13**	**11**	**7**	10	9
7	7	8	7	9	7	15	13	16	12	14	16	14	15	14	17	10

The optimum solution is then:

$$\mathbf{x}^{\text{opt}} = (2,16), \quad f\left(\mathbf{x}^{\text{opt}}\right) = f_{2,16} = 2. \qquad [1.19]$$

(The solution [1.19] is even the global one – read carefully the Table 1.3.)

The search does not end however like suggested in Table 1.3, as the algorithm always is seeking for a better solution. One can easily see that, if the procedure is continued, the optimum solution [1.19] cannot change and the algorithm exits from step 3.4.6.2, where the invariance of the best solution for N iterations is detected.

If the taboo list is limited to 10 elements, then, starting from the movement:

$$\begin{array}{c} (5,14) \xrightarrow{\text{right}} (5,15), \\ f=11 \qquad\quad f=7 \end{array} \qquad [1.20]$$

its length should be adjusted. Since the current optimal value of criterion is $f\left(\mathbf{x}^{\text{opt}}\right) = f_{5,9} = 6$, while the new solution performance is $f_{5,15} = 7$, the search procedure is at step 3.4.6.3 of Algorithm 1.5. Consequently, the solution $(5,9)$ (corresponding to the staring point) has to be removed from the taboo list. To compensate the loss, the new solution, $(5,15)$, is added to the list. This operation is not affecting the optimal solution though. The blocking counter is thus enforced to increment (see the step 3.4.6.1 of the algorithm).

The adjustment of taboo list continues in the same way, until the movement below:

$$\begin{array}{c} (4,15) \xrightarrow{\text{up}} (3,15), \\ f=8 \qquad\quad f=3 \end{array} \qquad [1.21]$$

when the optimal solution has to change from $f\left(\mathbf{x}^{\text{opt}}\right) = f_{5,9} = 6$ to $f\left(\mathbf{x}^{\text{opt}}\right) = f_{3,15} = 3$. Now, the procedure is at step 3.4.5.5 of the algorithm and the oldest two solutions of taboo list have to be removed. More specifically, the solutions $(3,9)$ (with $f_{3,9} = 10$) and $(3,10)$ (with $f_{3,10} = 12$) are the removed candidates. In turn, the new solution, $(3,15)$ (with $f_{3,15} = 3$), is added to the list.

1.4.7.2. *The problem of N queens*

The problem is to place N queens on a chessboard of size $N \times N$, such that no queen is threatening the others (two queens cannot lie on the same row, column or diagonal, otherwise they are in conflict to each other). The taboo search metaheuristic can be used to solve the problem.

In the following example, 7 queens are considered. Moreover, the taboo list length is set to 3. The only accepted movements are row permutations. In order to better understand how the taboo procedure works, beside the chessboard in current configuration, to the right side, a table shows all permutations leading to the minimum costs (i.e. number of conflicts), stating from that configuration. The Monte-Carlo procedure can be employed to generate possible movements, as their total number per iteration is:

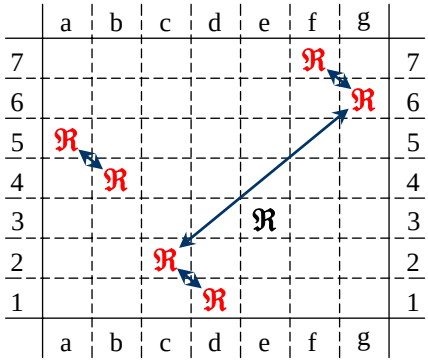
$$1+2+3+4+5+6 = \frac{6 \cdot 7}{2} = 21. \qquad [1.22]$$

In this application, the Algorithm 1.5 has slightly been modified: the taboo list does not include solutions any more (i.e. chessboard configurations showing the queens placement), but possible movements. In theory, this list can include up to 21 elements. (Writing this version of taboo search algorithm can be a useful exercise for the reader.) The movements leading to the taboo solutions are tagged by a star. Such movements actually become taboo as well.

In this example, the aspiration principle will be applied in the end. According to this principle, a taboo movement is allowed, provided that it leads to a better solution than the current one (and maybe to the optimum solution). Note however that the Algorithm 1.5 did not integrate this principle. Thus, after analyzing this example, perhaps the reader will be tempted to design another (more sophisticated) version of the Algorithm 1.5, so as to include the aspiration principle.

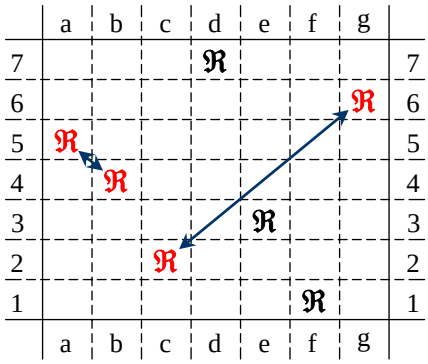
The taboo procedure iterations are described hereafter.

- Initial configuration (4 conflicts, as pointed by the arrows)



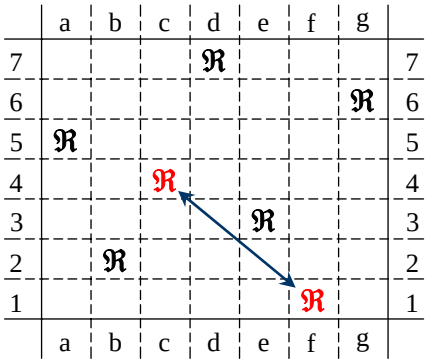
$\mathcal{M}_{0,0}$	Conflitcs	Cost
$1 \leftrightarrow 7$	a5 ⇌ b4 c2 ⇌ g6	2
$2 \leftrightarrow 4$	c4 ⇌ f7 f7 ⇌ g6	2
$2 \leftrightarrow 6$	a5 ⇌ b4 c6 ⇌ g2	2
$5 \leftrightarrow 6$	c2 ⇌ d1 e3 ⇌ g5	2
$1 \leftrightarrow 5$	c2 ⇌ g6 d5 ⇌ f7 f7 ⇌ g6	3

- Iteration no. 1 (after the movement $1 \leftrightarrow 7$ that becomes taboo; 2 conflicts)



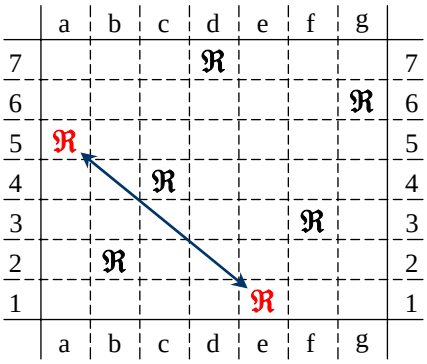
$\mathcal{M}_{1,1}$	Conflicts	Cost
$2 \leftrightarrow 4$	c4 ⇌ f1	1
$1 \leftrightarrow 6$	a5 ⇌ b4 e3 ⇌ g1	2
$2 \leftrightarrow 5$	b4 ⇌ c5 c5 ⇌ e3	2
$1 \leftrightarrow 2$	a5 ⇌ b4 c1 ⇌ e3 e3 ⇌ f2	3
$1 \leftrightarrow 3$	a5 ⇌ b4 b4 ⇌ e1 c2 ⇌ g6	3

➢ Iteration no. 2 (after the movement $2 \leftrightarrow 4$ that becomes taboo; 1 conflict)



$\mathcal{M}_{2,1}$	Conflicts	Cost
$1 \leftrightarrow 3$	$a5 \rightleftharpoons e1$	1
$6 \leftrightarrow 7$	$b2 \rightleftharpoons g7$ $c4 \rightleftharpoons f1$	2
$1 \leftrightarrow 7^*$	$c4 \rightleftharpoons f7$ $f7 \rightleftharpoons g6$	2
$2 \leftrightarrow 4^*$	$a5 \rightleftharpoons b4$ $c2 \rightleftharpoons g6$	2
$4 \leftrightarrow 5$	$a4 \rightleftharpoons d7$ $c5 \rightleftharpoons e3$	2

➢ Iteration no. 3 (after the movement $1 \leftrightarrow 3$ that becomes taboo; 1 conflict)

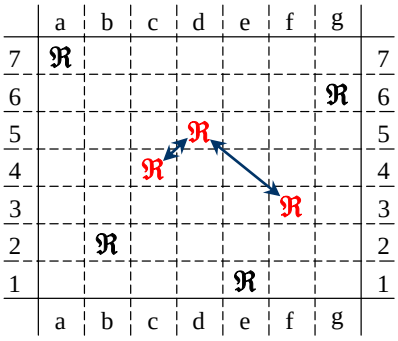


$\mathcal{M}_{3,1}$	Conflicts	Cost
$1 \leftrightarrow 3^*$	$c4 \rightleftharpoons f1$	1
$1 \leftrightarrow 7^*$	$d1 \rightleftharpoons f3$	1
$5 \leftrightarrow 7$	$c4 \rightleftharpoons d5$ $d5 \rightleftharpoons f3$	2
$6 \leftrightarrow 7$	$a5 \rightleftharpoons e1$ $b2 \rightleftharpoons g7$	2
$1 \leftrightarrow 2$	$b1 \rightleftharpoons g6$ $c4 \rightleftharpoons e2$ $e2 \rightleftharpoons f3$	3

➢ Iteration no. 4 (after the movement $5 \leftrightarrow 7$ that becomes taboo; 2 conflicts)

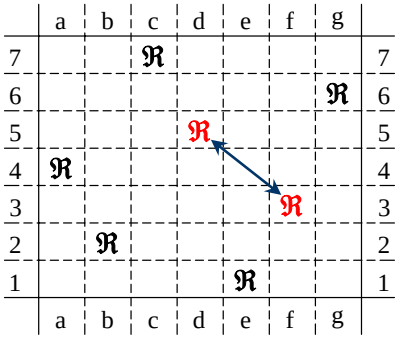
This iteration requires prior fulfillment of taboo principle. Consequently, the number of conflicts increased with respect to the previous iteration. The aspiration principle cannot be applied here, as no forbidden movement (the first two of $\mathcal{M}_{3,1}$) is leading to the optimum solution (without conflicts).

Since at the previous iteration a single conflict was obtained, the only acceptable better solution is the optimum one (with no conflicts).
The taboo list of overpasses here the limit of 3 movements: $1 \leftrightarrow 7$, $2 \leftrightarrow 4$, $1 \leftrightarrow 3$ and $5 \leftrightarrow 7$. It follows that the oldest movement, namely $1 \leftrightarrow 7$, has to be removed from the list, while conserving the 3 others.



$\mathcal{M}_{4,3}$	Conflicts	Cost
$4 \leftrightarrow 7$	d5 ⇌ f3	1
$5 \leftrightarrow 7^*$	a5 ⇌ e1	1
$1 \leftrightarrow 5$	b2 ⇌ e5 d1 ⇌ f3	2
$2 \leftrightarrow 5$	b5 ⇌ c4 d2 ⇌ e1	2
$2 \leftrightarrow 4$	b4 ⇌ e1 c2 ⇌ g6 d5 ⇌ f3	3

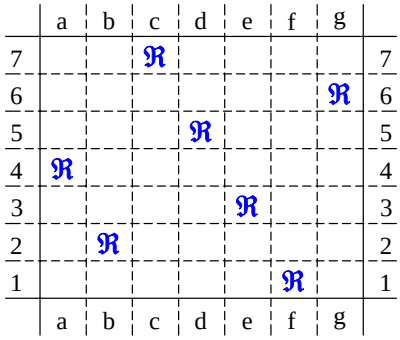
➢ Iteration no. 5 (after the movement $4 \leftrightarrow 7$ that becomes taboo; 1 conflict)



$\mathcal{M}_{5,1}$	Conflits	Coût
$1 \leftrightarrow 3^*$	–	0
$5 \leftrightarrow 6$	c7 ⇌ d6	1
$5 \leftrightarrow 7$	a4 ⇌ d7	1
$1 \leftrightarrow 6$	d5 ⇌ e6 d5 ⇌ f3	2
$1 \leftrightarrow 7$	b2 ⇌ c1 d5 ⇌ f3	2

Here, the movement $2 \leftrightarrow 4$ is removed from taboo list, being replaced by the movement $4 \leftrightarrow 7$.
When looking now at the table on right side, one notices that the movement $1 \leftrightarrow 3$ leads to the optimum solution (no conflicts between queens), despite its taboo label. Thus, the aspiration principle can now be applied, in order to complete the procedure.

➢ Iteration no. 6 (after the taboo movement $1 \leftrightarrow 3$; 0 conflicts)



1.5. Simulated annealing

1.5.1. *Principle of thermal annealing*

The optimization technique based on simulated annealing allows escaping form local optimums. It was inspired by the technique of controlled annealing, as employed in metallurgy. When a melted metal is cooled down very fast, a meta-stable solid is obtained. Moreover, this solid state corresponds to a local minimum of the internal energy. On the contrary, if the metal is cooled down slowly, the resulted crystalline structure corresponds to an absolute minimum of internal energy. If the temperature decreases too fast, the crystallization is not perfect. In this case, the imperfections can be attenuated by first reheating the metal, in order to release a part of the energy, and then cooling it again slowly. The succession of heating-cooling operations are founding the annealing phenomenon.

The modern annealing technique takes into account some properties that the final metallic product should have. Very often, this product is made of steel. In this case, two properties are absolutely necessary: elasticity and stiffness. Unfortunately, the two properties are opposite not only in terms of realization, but also as requirements applied to internal chemical composition of steel. As already known, the steel is obtained by mixing the iron with at least 2.11% carbon and other chemical substances from Mendeleyev table (such as Si, Mn, P, S, Cr, Ni, V, Ti, Mo). Depending on their proportions, the mixed substances can lead to a large panoply of steep properties, provided that the thermal annealing is carried out according to a carefully designed program. It is not the scope of this section to elaborate in direction of steel manufacturing. The only aim here is to outline a phenomenon related to annealing, which includes an intrinsic optimization mechanism, possibly to simulate on computer. If the liquid steel is cooled down too fast, then many chemical links between the molecules of its components cannot be established and internal tensions accumulate rapidly. As a result, the final product is very stiff, but breakable and inelastic. If, on the contrary, the liquid steel is cooled

down too slowly, then some of the chemical components (especially the carbon) are burned, even though they already established molecular links with other substances. In this case, the final product may be elastic, but too soft, without enough stiffness. In order to ensure a good trade-off between the elasticity and stiffness of final steel made product, it is thus necessary to design a "good" cooling down program, i.e. an optimal annealing procedure. This is an optimization problem based on two opposed criteria. Since in metallurgy such annealing programs already have been designed and implemented, there is a good chance to be able to simulate them in the framework of Optimizations.

The simulated annealing algorithm is actually reproducing a simplified process. The internal energy is here the criterion to minimize, whereas the temperature is a parameter directly related to the convergence. The algorithm principle is the following: as long as the temperature decreases, for each one of its values, the state of minimum internal energy is wanted. Such a state is offering the minimum amount of internal tensions, which constitutes the premise for a good trade-off elasticity-stiffness in the end. Like for other metaheuristics, this only is an intuitive interpretation of a natural phenomenon. The numerical procedure of simulated annealing can be applied to various optimization problems, regardless their correlation to the thermal annealing from metallurgy.

1.5.2. *Kirkpatrick's model of thermal annealing*

A model of thermal annealing largely accepted as being quite realistic was introduced by S. Kirkpatrick in [KIR 83].

The criterion to minimize, f, is referred to as *energy*. The energy is computed for states of a dynamical system. Starting from some optimal state $\mathbf{x}_i$ of energy $\mathcal{E}_i = f(\mathbf{x}_i)$, another neighbor state $\mathbf{x}_j$ can or cannot be selected as next optimal state depending on its energy, $\mathcal{E}_j = f(\mathbf{x}_j)$. If the energy decreases ($\mathcal{E}_j < \mathcal{E}_i$), then the $\mathbf{x}_j$ state replaces the $\mathbf{x}_i$ state. Otherwise, the $\mathbf{x}_i$ state can be replaced by the $\mathbf{x}_j$ state, only if some requirements are met.

If the energy variation $\Delta\mathcal{E}_{j,i} = \mathcal{E}_j - \mathcal{E}_i$ is positive, denote by:

$$\wp\left(\Delta\mathcal{E}_{j,i}, T\right) = \exp\left(-\frac{\Delta\mathcal{E}_{j,i}}{T}\right) \qquad [1.23]$$

the probability to obtain this variation at the temperature $T > 0$. When varying the normalized energy and the temperature, the probability surface of Figure 1.3, can be drawn. This image actually is the kernel of Kirkpatrick's model associated to annealing phenomenon.

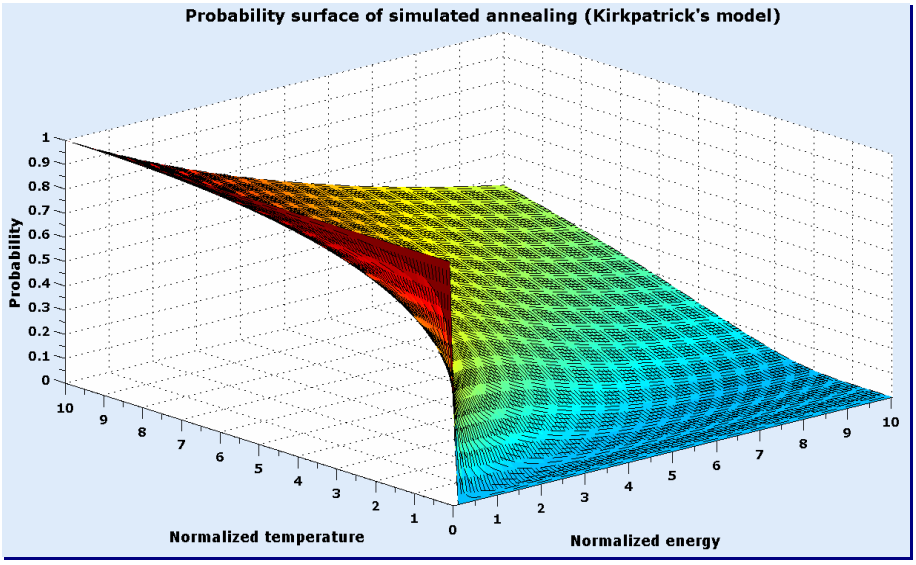


Figure 1.3. *Characteristic probability surface of annealing Kirkpatrick model.*

The shape of this surface is quite realistically describing the annealing phenomenon. Regardless the temperature, it is very likely that small energy differences exist between neighbor states. The probability to have big energy variation between states decreases more or less rapidly (see the topology of cliff over the temperature axis). For small temperatures, the probability is fast decreasing. The bigger the temperature, the smaller the energy decreasing rate (see the left-up corner of the surface). In other words, the big energy difference between neighbor states seemingly appears at high temperature and is less probable at low temperature.

Changing the probability surface means practically to change the heart of simulated annealing algorithm. This surface plays the same role for simulated annealing as the cooling program for the natural annealing.

To decide whether a state $\mathbf{x}_j$ of higher energy will replace or not the current state $\mathbf{x}_i$ (i.e. in case $\Delta\mathcal{E}_{j,i} = \mathcal{E}_j - \mathcal{E}_i > 0$), the probability surface can be employed. Thus, some U-PRSG is calibrated to generate a probability threshold $p_i \in (0,1)$, in order to compare it to the computed probability of Kirkpatrick model, [1.23]. If $\wp\left(\Delta\mathcal{E}_{j,i}, T\right) > p_i$, then $\mathbf{x}_j$ will replace $\mathbf{x}_i$. Otherwise, $\mathbf{x}_j$ cannot overthrown $\mathbf{x}_i$ and the search shall focus on a new state $\mathbf{x}_j$ in the neighborhood of $\mathbf{x}_i$. From the Probabilities Theory point of view, the event "$\mathbf{x}_j$ replaces $\mathbf{x}_i$" occurs with the probability $\wp\left(\Delta\mathcal{E}_{j,i}, T\right)$. (Since a U-PRSG was calibrated to generate numbers

within the $(0,1)$ interval, the probability to obtain a number below $\wp\left(\Delta\mathcal{E}_{j,i},T\right)$ is just $\wp\left(\Delta\mathcal{E}_{j,i},T\right)$.)

If no improvement is obtained after a certain number of iterations, the search is restarted in the same way, but after decreasing the temperature (T). To stop the search, one can count the number of successive temperature decreasing attempts while the optimization criterion still does not improve.

1.5.3. *Simulated annealing algorithm*

The main steps of simulated annealing procedure (based on Kirkpatrick model) are listed within the Algorithm 1.6.

Algorithm 1.6. *Simulated annealing procedure.*

1. Input data:
- Search vicinity $\mathcal{V}$ (equations allowing the user to decide whether a point belongs or not to this set).
- Energy f (criterion to minimize).
- Accuracy threshold, $\varepsilon > 0$.
- Maximum number of successive states for which the energy is not minimized any more, M.
- Maximum number of succesive temperature changing for which the current minimal state does not change, K.
- Maximum number of attempts to change the current minimal state, at constant temperature, N.
- Maximum temperature to test, T. (The successive temperatures are generated inside the interval $(\alpha T,T)$, where $\alpha \in (0,1)$ is a priori known.)

2. Initialization.

a. Select at random (but with uniform distribution) the starting state $\mathbf{x} \in \mathcal{V}$. A U-PRSG of nx size has to be used in this aim. If the starting state does not belong to $\mathcal{V}$, then it will be generated until this property is verified.

b. Evaluate the initial state energy: $\mathcal{E} = f(\mathbf{x})$.

c. Set the first minimal solution: $\mathbf{x}^{\min} = \mathbf{x}$ and $\mathcal{E}^{\min} = \mathcal{E}$.

d. Set a counter related to the number of successive temperature changes before replacing the current state (the *temperature counter*): $k = 0$.

e. Set a counter related to to the number of successive states that cannot really improve the energy beyond the accuracy threshold (the *energy counter*): $m = 0$.

f. Set a counter related to the number of attempts to change the minimal state at constant temperature (the *immobility counter*): $n = 0$.

3. Perform cooling down. For the current state $\mathbf{x}$:

3.1. Use a U-PRSG to generate an offset and a direction: $\Delta\mathbf{x}$. The generator has to operate in a hyper-cube including the vicinity, but as narrow as possible.

3.2. While $\mathbf{x} + \Delta\mathbf{x} \notin \mathcal{V}$, calibrate the U-PRSG to generate a new offset and direction, but inside the hyper-cube $[\mathbf{0}, \Delta\mathbf{x}]$, where $\Delta\mathbf{x}$ is the most recent vector offset.

3.3. Set the offset: the first generated $\Delta\mathbf{x}$, for which the vicinity limits are not violated. Thus, $\mathbf{x}^{\text{rep}} = \mathbf{x} + \Delta\mathbf{x} \in \mathcal{V}$ is the state that might replace the current state.

3.4. Evaluate the new state energy: $\mathcal{E}^{\text{rep}} = f\left(\mathbf{x}^{\text{rep}}\right)$.

3.5. Compute the energy difference with respect to the current minimal state: $\Delta\mathcal{E} = \mathcal{E}^{\text{rep}} - \mathcal{E}^{\min}$.

3.6. If $\Delta\mathcal{E} < 0$, then:

3.6.1. Replace the current minimal state and energy: $\mathbf{x}^{\min} \leftarrow \mathbf{x}^{\text{rep}}$ and $\mathcal{E}^{\min} \leftarrow \mathcal{E}^{\text{rep}}$.

3.6.2. Replace the current solution: $\mathbf{x} \leftarrow \mathbf{x}^{\text{rep}}$ and $\mathcal{E} \leftarrow \mathcal{E}^{\text{rep}}$.

3.6.3. Reset the temperature counter: $k \leftarrow 0$.

3.6.4. Reset the immobility counter: $n \leftarrow 0$.

3.6.5. If $\Delta\mathcal{E} > -\varepsilon$, then:

3.6.5.1. Increment the energy counter: $m \leftarrow m + 1$.

3.6.5.2. If $m > M$, stop the search, as no real improvement is made. Go directly to the final stage (no. 4).

3.6.6. Otherwise, reset the energy counter: $m \leftarrow 0$.

3.7. Otherwise, the energy of the possible new solution is at least equal to the energy of the current solution and then:

3.7.1. Increment the immobility counter: $n \leftarrow n + 1$.

3.7.2. If $n \le N$, then:

3.7.2.1. Compute the energy difference with respect to the current state: $\Delta\mathcal{E} = \mathcal{E}^{\text{rep}} - \mathcal{E}$.

3.7.2.2. If $\Delta\mathcal{E} < 0$, replace the current solution: $\mathbf{x} \leftarrow \mathbf{x}^{\text{rep}}$ and $\mathcal{E} \leftarrow \mathcal{E}^{\text{rep}}$.

3.7.2.3. Otherwise:

a. Compute the Kirkpatrick probability: $\wp\left(\Delta\mathcal{E}, T\right)$ (by using definition [1.23]).

b. Use a U-PRSG to generate a number $p \in (0,1)$.

c. If $\mathcal{P}(\Delta\mathcal{E},T) > p$, then replace the current solution: $\mathbf{x} \leftarrow \mathbf{x}^{\text{rep}}$ et $\mathcal{E} \leftarrow \mathcal{E}^{\text{rep}}$. Otherwise, keep the current solution.

3.7.3. Otherwise, the temperature has to be decreased:

3.7.3.1. Use a U-PRSG to generate a new temperature, a smaller one, in the interval $(\alpha T, T)$. The new temperature replaces the current temperature, T.

3.7.3.2. Reset the immobility counter: $n \leftarrow 0$.

3.7.3.3. Increment the temperature counter: $k \leftarrow k+1$.

3.7.3.4. If $k > K$, stop the search, as the minimal solution cannot change any more. Go directly to the final stage (no. 4).

3.8. Resume the search from step 3.1.

4. Return:

- The minimal current state: $\mathbf{x}^{\min}$.
- The current minimal energy: $\mathcal{E}^{\min} = f\left(\mathbf{x}^{\min}\right)$.

This algorithm has been designed so as to avoid infinite loops. The two main counters (of temperature k and of immobility n) are in charge with preventing the procedure to be captured in such a loop. Like in case of previous algorithms (or the following ones) there is a number of configuring parameters that have to be specified before initiating the run. Usually, the user doesn't know how to set such parameters. Therefore some preliminary running tests are necessary in order to obtain a fine tuning of the algorithm. Normally the maximum number of states which are not changing the optimal solution at the same temperature, N, varies from 50 to 200. The maximum number of temperature change, K, generally is smaller, between 5 and 25. The maximum number of states for which the energy does not actually improve, M, is even smaller, between 1 and 10. The accuracy threshold ε depends on the criterion representation scale, as usual.

The initial temperature, T, has to be sufficiently large, in order to avoid capturing in a local minimum. The temperature decreasing strategy employed within the Algorithm 1.6 is not unique. (The Monte-Carlo principle actually was implemented here.) Different strategies can be adopted, depending on each application particularities. In any case, such a strategy should have a unique main goal: to increase the chance of global minimum to be found, in the given vicinity.

Finally, the decreasing parameter α is set between 0.8 and 0.9.

Another way to reduce the temperature is to split the $(0,T)$ interval into several smaller segments (uniform or not) and to operate with their bounds.

1.6. Tunneling

1.6.1. *Tunneling principle*

The main purpose of Tunneling algorithms is to allow escaping from the local optima [LEV 85], [BAR 91], [HAM 05], [GHE 07]. Their principle is quite simple: each time the search has reached a local optimum, a tunnel is carven, in order to find a valley in the criterion variation, which could lead to another optimum. The principle is illustrated in Figure 1.4.

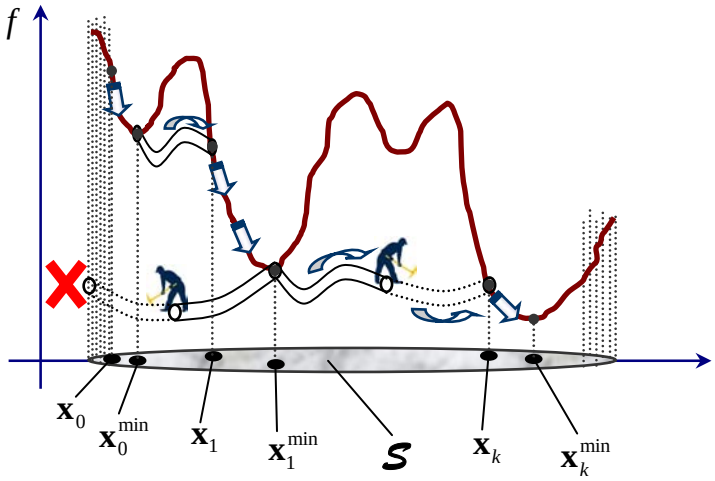


Figure 1.4. *Illustration of tunneling principle.*

As usual, assume the criterion has to be minimized. The procedure starts with a minimization phase initiated from a point $\mathbf{x}_0$. After reaching the local minimum $\mathbf{x}_0^{\min}$, a tunneling phase follows. The search space is thus pierced until another initialization is found, $\mathbf{x}_1$. The local search is thus relocated on the attraction zone of another minimum, $\mathbf{x}_1^{\min}$. This strategy is repeated until one of the stop tests is verified. If the carved tunnel touches or trespasses the search space (or some vicinity) before finding the next valley, then the search is aborted and a new tunnel is carven, following a different direction.

Thus, basically, a tunneling algorithm includes two phases:

1. Local minimization on a valley (the *exploiting phase*).
2. Migration towards another search zone by tunneling, in order to find another valley to exploit (the *tunneling phase*).

In the exploiting phase, a local optimization method can be invoked. Thus, starting from the initial solution $\mathbf{x}_k$ ($k \in \mathbb{N}$), a local minimum $\mathbf{x}_k^{\min}$ is found (as

shown in Figure 1.4). In the tunneling phase, actually, a new initial solution is wanted, at the end of the carved tunnel.

1.6.2. *Types of tunneling*

If the tunneling directions are selected at random, with the help of some U-PRSG, the search can be time consuming. During the search, the main issue is to avoid the already exploited zones. Therefore two types of tunneling usually are considered for implementation: stochastic and with penalties.

1.6.2.1. *Stochastic tunneling*

This technique was introduced in [WEN 99]. The main idea is to apply a non linear transformation to the criterion, in order to better isolate the visited valleys from the non visited ones. For example, here is such a transformation:

$$f(\mathbf{x}) \leftarrow f_{\gamma,k}^{\text{sto}}(\mathbf{x}) = 1 - \exp\left[-\gamma\left(f(\mathbf{x}) - f\left(\mathbf{x}_k^{\min}\right)\right)\right], \quad \forall \mathbf{x} \in \boldsymbol{S}, \qquad [1.24]$$

where: $\mathbf{x}_k^{\min}$ is the current minimal point, the best one found up to the k-th iteration; $\gamma > 0$ is a parameter defining the deforming degree of search space (unit, by default).

This function conserves the original placement of the already discovered minimum points and changes each time a better solution is found. However, the criterion is scaled differently. Thus, all the points leading to values larger than the current performance $f\left(\mathbf{x}_k^{\min}\right)$ are enforcing the criterion to be projected into the $[0,1]$ interval. This allows the search procedure to easily detect the initializations that seemingly are not of worth to be considered. (Some of them may be employed though, as they could be located on deeper valleys than the current one.) On the contrary, the criterion becomes negative (varying within $(-\infty,0)$) for the points with better performance than $f\left(\mathbf{x}_k^{\min}\right)$. Thus, the zones that have not yet been explored can easily be detected. Now, the goal is to perform exploitation mainly around the points with negative values of criterion [1.24].

1.6.2.2. *Tunneling with penalties*

According to this approach, the tunneling phase take benefit from some penalties applied on the performance of some points. For example, in case of traveling salesman problem, the distance between some cities can artificially be increased, which yields escaping from some local minimums and enlarging the search horizon. Also, penalties allow performing the search in virgin, unvisited zones.

For this type of tunneling, the penalties have to be defined according to each problem to solve. There is no general rule to set penalties and therefore only the principle can be stated in this context.

1.6.3. *Tunneling algorithm*

The procedure integrated into the Algorithm 1.7 is based on stochastic tunneling.

Algorithm 1.7. *Stochastic tunneling procedure.*

1. Input data:
- Search space $\mathcal{S}$ (equations allowing the user to decide whether a point belongs or not to this set).
- Criterion f to minimize.
- Parameter of search space deforming, $\gamma > 0$ (by default, $\gamma = 1$).
- Accuracy threshold $\varepsilon > 0$ (for local minimization).
- Maximum number of initializations to restart the local search, K.
- Maximum number of attempts to escape from the attraction of a minimal point, M.

2. Initialization.

a. Select at random (but with uniform distribution) the first departure point $\mathbf{x}_0 \in \mathcal{S}$. A U-PRSG of nx size has to be used in this aim. If the starting point does not belong to $\mathcal{S}$, then it will be generated until this property is verified.

b. Evaluate the performance of selected initial point: $f(\mathbf{x}_0)$.

c. Set a counter related to the number of tested initializations (the *initializations counter*): $k = 0$.

d. Set a counter related to the number of attempts to escape from the attraction of a minimal point (the *escape counter*): $m = 0$.

3. For each initial point,i.e. for $k = \overline{0, K-1}$:

3.1. Exploiting phase. Call a local optimization procedure (exact or metaheurisitc), in order to find a local minimum $\left\{\mathbf{x}_k^{\min}, f\left(\mathbf{x}_k^{\min}\right)\right\}$, with ε accuracy, starting from the initial point $\mathbf{x}_k$.

3.2. Stochastic tunneling phase. For $m \geq 0$:

3.2.1. Select the tunnel departure: $\mathbf{x}_{k,0} = \mathbf{x}_k^{\min}$.

3.2.2. For $n \geq 0$:

3.2.2.1. Use a U=PRSG of nx size to generate a vector offset $\Delta\mathbf{x}_{k,n+1}$. The offset length has to be sufficiently small (with respect to the search space diameter).

3.2.2.2. Define the next point on the tunnel path:

$$\mathbf{x}_{k,n+1} = \mathbf{x}_{k,n} + \Delta\mathbf{x}_{k,n+1}.$$

3.2.2.3. If $\mathbf{x}_{k,n+1}$ falls outside the search space ($\mathbf{x}_{k,n+1} \notin \mathcal{S}$),

then the tunnel is abandoned:

a. Increment the escape counter: $m \leftarrow m+1$.

b. If $m > M-1$, stop the search, as the search cannot escape from the attraction of the current minimal point. Go directly to the final stage (no. 4).

c. Otherwise, continue at step 3.2.1.

3.2.2.4. Otherwise, use the stochastic criterion [1.24] to test whether a new zone (not yet explored) has been discovered or not.

a. If $f_{\gamma,k}^{\text{sto}}\left(\mathbf{x}_{k,n+1}\right) < 0$, then a new local minimization has to be carried out. In this aim, set the initial point ($\mathbf{x}_{k+1} = \mathbf{x}_{k,n+1}$) for the local minimization procedure and jump to step 3.1. This is allowed only if, after incrementing the initializations counter k, the upper limit K is not violated. Otherwise the search has to be stopped and the current optimal solution has to be returned – see the final stage, no. 4).

b. Otherwise, continue to carve the tunnel: $n \leftarrow n+1$.

4. Return:

- The current minimal point: $\mathbf{x}_k^{\min}$.
- The current performance: $f\left(\mathbf{x}_k^{\min}\right)$.

1.7. GRASP Methods

The abbreviation of this methods group comes from: *Greedy Randomized Adaptive Search Procedure(s)*. The concept was introduced by T. Feo and M. Resende in [FEO 95] and actually reveals a principle that already has been suggested by the previous metaheuristic (the tunneling one). Thus, local optimization is performed, starting from a number of initial solutions, after being generated at random. The procedure is *adaptive* in the sense that the selection of new initial solutions takes into account the results of previous iterations. Although for the local optimization, any algorithm can be employed, one aims to choose the most efficient procedures. Nevertheless, the overall procedure usually is greedy. In turn, there is a low risk to be captured by a local optimum, thanks to multiple re-runs, from various initializations.

The reader may find interesting to generalize the previous metaheuristics according to GRASP principle. Nowadays, the modern optimization makes use of

several techniques in a same algorithm, in order to solve complex or difficult problems. Exact methods [BOR 13] and metaheuristics can be combined together, if possible. Usually, in this attempt, one starts from low complexity procedures to find the initial solutions. Then, more sophisticated procedures are employed to refine the search around each initialization, until the optimal solution is found (with prescribed accuracy). During the first optimization phase, the convergence speed prevails, as the accuracy is not of real concern. One aims here to rapidly find some attraction centers around which local exploiting zones can be delimited. In the next phase, the accuracy becomes more important than the convergence speed.

Overall, applying the GRASP principle, seemingly is the most successful way to deal with modern optimization problems.

Chapter 2

Metaheuristics – Global Methods

2.1. Principle of evolutionary metaheuristics

Charles Darwin's theory, as published in [DAR 1859] and especially in [DAR 1871], has attracted the interest of the worldwide scientists. In the world of optimization, this theory became a source of inspiration approximately 100 years after its publication. According to Darwin, life is characterized by two specific features: evolution and adaptation. Evolution is a long-term process, along several generations, which leads to an improved way of life. Moreover, evolution is a phenomenon observed within the population of living entities. Adaptation is rather affecting every living entity and characterizes its ability to change the way of living on a short term, depending on the natural and social environmental conditions. Here, the goal is to survive, despite the hostility of the environment, or to find the best way to survive. Both phenomena constitute the basis of a mechanism that Darwin referred to as *natural selection*. In fact, it is the mechanism that encodes optimization. Throughout this chapter, it will be shown how natural selection can be used to solve granular optimization problems.

Natural selection is presented at both microscopic and macroscopic levels simultaneously. In the first case, the entity affected by this mechanism is the living cell, or the *chromosome*. In the second case, it represents the individuals of certain living populations. By natural selection, only the cells or the most advanced and adaptable individuals are able to survive and transmit their genetic heritage for future generations. Nevertheless, while the other cells or individuals are gradually removed, they could contribute at the maintaining of a certain diversity. Without this diversity, the evolution would be affected by degenerative phenomena.

Natural selection is permanently acting around us. It can be found during the migration of animals in search for a better environment and even in big cities,

among humans, when looking for a better job. Globally, life is evolving towards its best, to an optimum. Therefore, it is interesting to simulate the evolution manner of the living entities, in order to solve some optimization problems. For this reason, heuristic methods described in this chapter are referred to as *evolutionary* or *by strategy of evolution*.

The granular optimization problems like [1.2], where the optimization criterion [1.1] has the already mentioned characteristics (fractality, granularity), are stated in this chapter as well. In context of evolutionary metaheuristics, this criterion, often called *fitness*, has to be maximized. Unlike the methods presented in the previous chapter, here, searching for the optimum is performed in parallel, by using a population of entities within the searching space, which evolve according to a prescribed strategy. The names of the entities correspond to an evolution strategy (chromosomes, particles, ants, fireflies, bees, etc.). As a result, the metaheuristics in this chapter are *global*, meaning that the searching space is simultaneously investigated by several explorers, which are designed to locate the global optimum.

The methods in this chapter are grouped into two categories: microscopic (related to *genetic algorithms*) and macroscopic (that simulate the evolution of a certain type of living beings like *ants*, *bees*, *fireflies* or individuals of a dynamic population with both cognitive and social conscience).

2.2. Genetic Algorithms

2.2.1. *Biology breviary*

First optimization concepts related to genetic mechanisms were introduced by John Holland in 1975 [HOL 75] (a second edition of this book being published in 1992, see [HOL 92]). Also, John Holland introduced the term *genetic algorithm*. Thus, a *genetic algorithm* (GA) is a technique for simulating the natural process of microscopic evolution and adaptation specific to biological systems. This definition has been refined and generalized by many scientists after Holland (see, for example, [RAW 91], [RAW 93], [KOZ 92], [GOL 94], [BAE 00]). An excellent study on modern GA and their applications (very appreciated by John Holland) was published in 1995 by Melanie Mitchell [MIT 95]. (The same book includes an impressive bibliography.)

The design of GA uses a vocabulary borrowed from natural genetics. In order to ease the understanding of the GA, it is useful to present the generally accepted definitions of key concepts in this vocabulary, which are grouped into a biological breviary.

- *Chromosome*: a chain of DNA molecules (deoxyribonucleic acid) that are found in organic cells, which encode the biological identity specific to a certain living organism.
- *Gene*: a functional bloc of a chromosome that encodes a specific protein. Each gene defines a characteristic of the living organism, including humans:

height, eye color, nose shape, ear length etc. The position of a gene in the chromosome is extremely important. By modifying this position one could produce significant changes in the basic characteristics of the body.

- *Allele*: a characteristic which is encoded by a gene. For example, eye alleles include all the colors a human eye can have. One can say that the allele is the *value* of a gene.
- *Genome*: a collection of genes and chromosomes of an organism.
- *Genotype*: a particular set of genes which belong to a genome.
- *Phenotype*: the set of mental and physical characteristics of an organism, at the beginning of its existence. For example: brain volume, intensity of native intelligence, eye color, etc.
- *Diploid*: an organism with an even number of chromosomes in each cell. Most organisms are diploid. Humans have 46 chromosomes in each cell. The main feature of diploids is the ability to multiply themselves (to reproduce) by crossing pairs of chromosomes.
- *Haploid*: an organism with an odd number of chromosomes in each cell. Some hermaphroditic organisms are haploid. For haploid organisms the reproduction only is possible by means of mutations or inversion of parts of a chromosome.
- *Population*: a collection of chromosomes (at microscopic level) or organisms who live at the same time in a certain environment (at macroscopic level).
- *Generation*: a structure of a population at specific time instant. One can say that populations evolve through generations by means of natural selection. This implies that some individuals in the population cease to exist starting from a particular generation, while other individuals begin their life in a certain generation.
- *Parents*: chromosomes or organisms involved in the reproduction of new individuals in a population.
- *Offspring (children)*: chromosomes or organisms resulted from reproduction. Since the parents are transmitting some of their characteristics to offspring, the latter are also known as *inheritors*.
- *(Natural) selection*: mechanism of population renewal across generations, so that the most capable and adaptable individuals (characterized by large values of the fitness) remain alive, while the others cease to exist.
- *Viability*: the probability of a chromosome or organism to survive and reproduce.
- *Fertility*: the number of generated offspring by a chromosome or organism during its lifetime.

The fitness is often considered a measure of the viability or fertility of an individual in the population, which justifies its use in the context of natural selection mechanism.

2.2.2. *Features of Genetic Algorithms*

Practically, GA are optimization procedures based on genetic and natural selection mechanisms. They use the principles of the strongest survivor (i.e. the most capable one) as well as the exchange of random information, which are sometimes guided, in order to build an evolution process that has certain characteristics ranging from exploration to exploitation. In fact, GA are iterative procedures for global search, whose goal is to maximize the fitness.

In the framework of GA, the population consists of particular chromosomes (or *individuals*, in general), each of which corresponding to an encoded representation of a potential solution for the granular problem. A chromosome is composed by genes that can take several values grouped in alleles. In the simplest form of GA, a chromosome consists of a binary string, represented by a sequence of 0 and 1; in more evolved forms, the chromosome is composed by sets of alphanumeric symbols (in which case it is sometimes referred to the general term of *individual*).

2.2.2.1. *Genetic operations*

The GA principle relies on the evolution of chromosomes population that can be realized by means of three (genetic) operations: crossover, mutation and inversion. Each operation is applied according to a predefined or variable probability. The chromosomes involved in a genetic operation are selected according to some selection rules a priori known (as discussed later).

Crossover is a genetic operation that enables the exchange of genes between two parents, as shown in Figure 2.1.

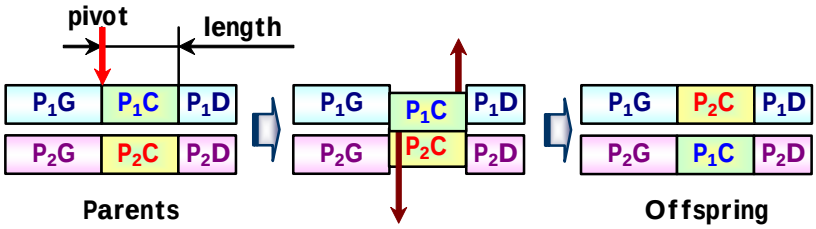


Figure 2.1. *Illustration of simple crossover between two chromosomes.*

The resulting offspring incorporate genes from both parents. The crossing area is determined by two parameters: *the pivot*, which indicates the position of the gene in the chromosome and *the length* (minimum unit), which sets the number of positions involved in the crossover operation. Usually, the two parameters are randomly chosen, but according to a certain probability density. The U-PRSG or P-PRSG are used in order to specify the parameters of the crossover at each iteration. (See Appendices A and B for *pseudo-random sequence generators* (PRSG).) In general,

the crossover rate corresponds to the proportion of individuals that are crossed, in order to create the next generation.

The crossover between chromosomes can be generalized, for example, by considering multiple pivots and lengths, randomly chosen. This provides a *multiple pivots* crossover. Another generalization leads to a *masked* crossover. The mask is a virtual binary chromosome, whose unit values indicate the positions to be exchanged in both parents. An example of mask crossover is shown in Figure 2.2. Usually, the pivots are randomly selected (in number and positions), using some U-PRSG.

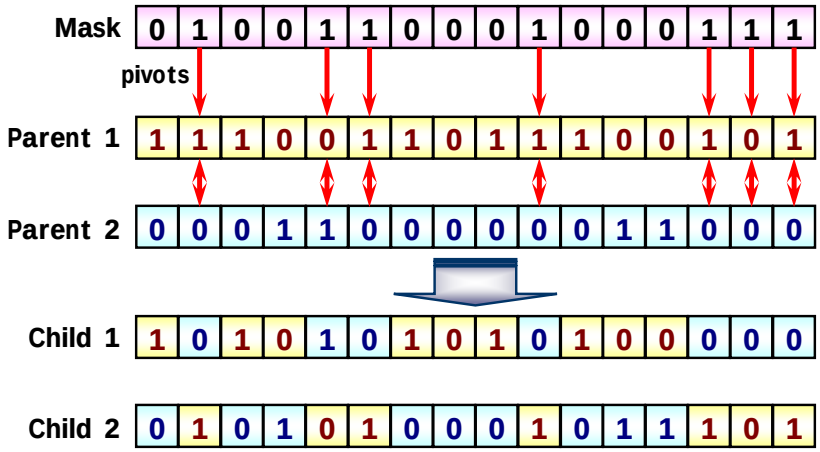


Figure 2.2. *Example of masking crossover between two chromosomes.*

Mutation is a random change of one or more alleles in a chromosome. It aims to prevent the pauperization of the population genetic heritage. The mutated individuals are rarely better than the initial ones and the mutation rate is low, if compared to the crossover rate. But these individuals can be involved in crossover operations with much more capable individuals, in order to produce offspring that are located quite away from the current search zone. Thus, the attraction of local optimum can be avoided. In reasonably small proportion, the mutants are benefic to preserve certain diversity of the population, which avoids its rapid degeneration to a local optimum. On the contrary, too much diversity determines the inability of the population to focus on the global optimum (if found) and can lead to oscillations. Therefore, it is suitable that the population stays focused enough and, at the same time, keeps sufficient diversity.

Usually, the mutation is performed by using a mask that indicates the start of the genes that can change the alleles, as suggested by Figure 2.3. In this figure, one can easily notice that the mutation consists of arrows changing within the mutated genes.

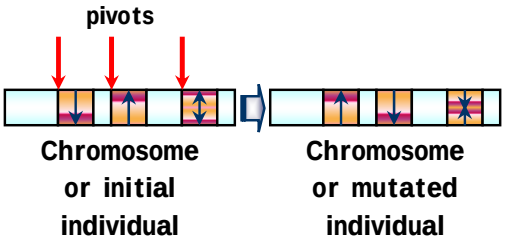


Figure 2.3. *Effect of mutation in a chromosome.*

The mask is either selected at random or according to chromosome specific internal structure (if a priori known). Usually, the lengths of the genes affected by mutation have to be known in advance, unless they are randomly chosen. Likewise, the mask may simply indicate certain atomic positions involved in the mutation, as in the example of Figure 2.2 (where all the genes have unit length).

Another way to apply mutation is to perform permutations between pairs of genes, like in the example of Figure 2.4.

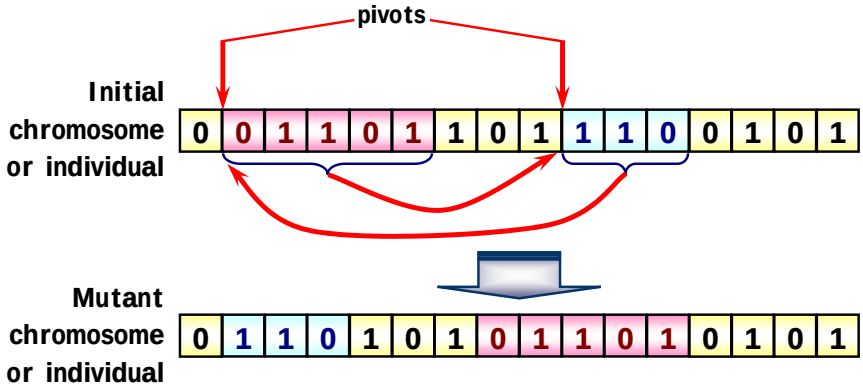


Figure 2.4. *Example of permutation.*

Inversion is a genetic operation that produces a change in the concatenation of the genes in a certain area of chromosome, so that the new gene sequence (series) is inverted with respect to the initial sequence. Figure 2.5 illustrates the principle of mutation. This operation is determined by the pivot that indicates the starting position of reversing area and the length of this area. Usually, the two parameters are randomly chosen. From the principle illustrated in Figure 2.5, one can easily imagine a more general inversion operation, with multiple pivots (as in the case of multiple crossing).

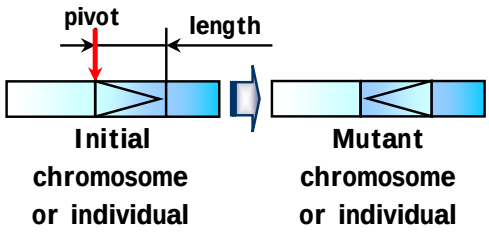


Figure 2.5. *Effect of inversion in a chromosome.*

The result of inversion is a mutant, like in case of mutations. (In some publications, the inversion actually is seen as a type of mutation.) Consequently, the rate of this type of mutants within the population must be small enough.

Normally, diploids are involved in crossover, while the mutations and the inversions are genetic operations characteristic to haploids. However, the crossover operation constitutes the engine of evolution, since the resulting inheritors are often better than their parents in terms of fitness. But, if used alone, crossover can produce individuals that become dominators in their population, which can slow down (or even block) the evolution. This yields the orientation of population towards a local fitness optimum, which becomes too strong as attraction point. To escape from such attraction, it is necessary to take drastic measures on the population, such as, for example, to apply crossover between very capable individuals and mutants (produced by mutation or inversion). Thus, the population diversity is increased and a different optimum of fitness can be approached in a different zone of search space. If the attraction point represents the global optimum of fitness, then each time the population will return in its vicinity.

Genetic operations are applied at random, but each one has its own probability. In general, the crossover probability (P_c) is significantly higher than the mutation probability (P_m) or inversion probability (P_i). In addition, these probabilities may vary from one generation to another. For example, usually, $P_c \in [0.5, 0.95]$ and $P_m, P_i \in [0.005, 0.01]$.

2.2.2.2. *Inheritors viability*

An important issue to be considered when a genetic operation is carried out is the viability of inheritors (offspring or mutants). These operations cannot guarantee that the inheritors remain in the search space, which is a compulsory condition to be kept alive. To better understand this problem, two examples are given.

Assume, for example, that the search space is the set of natural number $\overline{0,50\,000}$ and the generic chromosome is the binary representation of these numbers, which contains 4 successive genes of length 4 each. Thus 16 bits for each chromosome are

necessary. The chromosome corresponding to the upper limit of the search space (50 000) is: 1100 0011 0101 0000. A simple permutation of the genes in the middle of chromosome produces the mutant: 1100 0101 0011 0000, which leads to the number 50 480 , outside the search space. If the inversion is applied to the second gene (0011), then the 1100 1100 0101 0000 mutant is obtained, which represents the number 52 304 – also outside the search space. Finally, a crossover with the chromosome 0000 0000 0000 1111 (i.e. the number 15), which requires the exchange between the last genes, produces two offspring: 1100 0011 0101 1111 (the number 50 015) and 0000 0000 0000 0000 (the number 0). Certainly, the first child is not viable.

A second example is related to the traveling salesman problem, described in the previous chapter (see section 1.3.3). Recall that the traveler must pass through each one of the N cities only once, by minimizing the journey distance. Here, the chromosome consists of the sequence of visited cities in a certain order (each one represented by a letter). Then, after a sequence of genetic operations (especially crossover), offspring often become non-viable: a city can be visited several times and another can never be visited. Figure 2.6 shows such a case, where the first child requires the cities A and F to be visited twice, while the cities H and G are not visited at all. The second child is not viable either, because this time, the cities A and F are not visited, while the cities H and G must be visited twice.

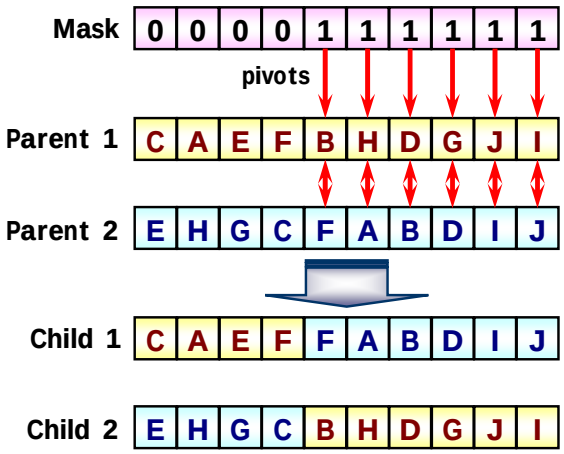


Figure 2.6. *Example of non-viable crossover in the traveling salesman problem.*

The examples above show that the result of a genetic operation should be verified before proceeding to the next step. It is necessary to work with viable chromosomes only. To avoid wasting the search time, it is wise not to remove the non-viable inheritors, but rather to make them viable, which means to bring them

back into the search space. There is no general definition for this operation. The making viable approach always has to be adapted to the nature and/or structure of the search space.

Come back to the previous two examples. If the search space has numerical limits (such as 0 and 50000 in the first example), then the inheritors can become viable through the modulo operation. Each non-viable inheritor may be improved by computing the remainder of the division by a number determined from the boundaries of the search space. In the first example, it suffices to replace the non-viable inheritor h with $h\%50001 \in \overline{0,50000}$ (where $n\%N$ is the reminder of the division between n and N, i.e. $n \bmod N$). Thus, the first mutant 50480 becomes 479 (0000 0001 1101 1111), the second mutant 52304 is replaced by 2303 (0000 1000 1111 1111), whereas the child 50015 is transformed into 14 (i.e. 0000 0000 0000 1110). The modulo operation is not the only instrument to make inheritors viable, but its main advantage is the small computational burden.

In the case of traveling salesman, the viability could be obtained, for example, as follows: in the chromosome, the first city encountered for the second time is replaced with the first missing city and so on. With this strategy, the two non-viable offspring from Figure 2.6 are replaced by viable ones, namely CAEF**HG**BDIJ and EHGCB**A**D**F**JI, respectively.

2.2.2.3. *Selection for reproduction*

In the framework of GA, two types of selection are necessary: a selection of individuals (chromosomes) to be involved in reproduction by genetic operations and a selection of individuals who will survive within the population. In this section, some techniques for the first type of selection (for reproduction) are described.

The manner of selecting the individuals who are appropriate for the reproduction can decisively influence the evolution of the population. Two extreme situations can be observed:

a. If the selection is too strict (only few individuals of the population are aptly for reproduction), then there is the risk that the population becomes dominated by sub-optimal individuals, which, moreover, tend to reduce its diversity.
b. If the selection is too permissive (too many low quality individuals are involved in reproduction), then the population could become too scattered and its evolution towards an optimum is too slow (or oscillating).

Therefore, one has to realize a good trade-off between the diversity of the population (the *exploration ability*) and the evolution speed (the *exploitation capacity*). Choosing the most appropriate selection manner remains an open problem in the framework of GA. Nevertheless, there are some selection techniques that have proven their efficiency in many applications. Some of them are described next.

Denote the population of chromosomes or individuals by $\mathcal{P}$. Certainly, at any time of evolution, the population contains N viable individuals ($\mathcal{P} \subset \mathcal{S}$).

A. Selection by fitness

Since the fitness can be evaluated for each individual in the $\mathcal{P}$ population, it is natural to consider that this criterion is a direct measure of reproduction capacity [HOL 75]. More specifically, one says that the individual $\mathbf{x} \in \mathcal{P}$ is *more suitable (more fitted)* for reproduction than the individual $\mathbf{y} \in \mathcal{P}$ if $f(\mathbf{x}) > f(\mathbf{y})$. Consequently, a probability density of reproduction associated to the population $\mathcal{P}$ can be constructed by normalization, as follows:

$$p(\mathbf{x}) = \frac{f(\mathbf{x})}{\sum_{\mathbf{y} \in \mathcal{P}} f(\mathbf{y})}, \quad \forall\, \mathbf{x} \in \mathcal{P}. \qquad [2.1]$$

Once this probability density has been set, it suffices to use the Baker's algorithm (BA) in Appendix B (a P-PRSG in fact), in order to select the individuals for reproduction (by crossover, mutation or inversion). To increase the accuracy of this algorithm, it is useful to replace the probability density [2.1] by the histogram below:

$$p(\mathbf{x}) = N \frac{f(\mathbf{x})}{\sum_{\mathbf{y} \in \mathcal{P}} f(\mathbf{y})}, \quad \forall\, \mathbf{x} \in \mathcal{P}. \qquad [2.2]$$

Thus, even the individuals with reduced reproduction capabilities have a chance to be selected (see the Baker's generalized algorithm (BGA) in Appendix B.)

It is self understood that the selection profile managed by the fitness changes at each generation, knowing that some individuals in the population are replaced with their inheritors.

Despite the simplicity and the natural manner of selection based on fitness, there is a major disadvantage though. Initially, the dispersion of the population according to fitness is large enough, since there are few suitable individuals for reproduction and many individuals with reduced fitness. As the population evolves, there is a risk that the capable individuals quickly become to dominate the population. This results in a premature convergence to a local optimum, caused by excessive exploitation, to detriment of exploration. Generally, this type of selection too much accentuates the exploitation, while limiting the exploration ability of the population.

B. Selection by σ-normalization

This technique aims to reduce or to avoid the phenomenon of premature convergence. The idea is to maintain the selection rate of fertile individuals

at an approximately constant level, throughout the evolution of the population [FOR 85]. (With the previous selection technique, the rate can rise very quickly.) To do so, it is necessary to use the average fitness and the population variance:

$$\mu(\mathcal{P}) = \frac{1}{N}\sum_{\mathbf{x}\in\mathcal{P}} f(\mathbf{x}) \; ; \quad \sigma(\mathcal{P}) = \sqrt{\frac{1}{N}\sum_{\mathbf{x}\in\mathcal{P}} \left(f(\mathbf{x}) - \mu(\mathcal{P}) \right)^2} \; . \qquad [2.3]$$

The reproduction capacity of the individuals in the population is then defined as follows (which justifies the term of *σ-normalization*):

$$r(\mathbf{x}) = \begin{cases} 1 & , \text{ if } \sigma(\mathcal{P}) = 0 \\ 1 + \dfrac{f(\mathbf{x}) - \mu(\mathcal{P})}{2\sigma(\mathcal{P})} & , \text{ if } \sigma(\mathcal{P}) > 0 \end{cases} , \quad \forall \mathbf{x} \in \mathcal{P} \; . \qquad [2.4]$$

For the unable individuals who have to be involved in the reproduction, $r < 0$, which involves $f < \mu(\mathcal{P}) - 2\sigma(\mathcal{P})$. Thus, there is a selection threshold defined by $\mu(\mathcal{P}) - 2\sigma(\mathcal{P})$. If the threshold is too tight or too large, the user can modify the definition [2.4], as follows:

$$r(\mathbf{x}) = \begin{cases} 1 & , \text{ if } \sigma(\mathcal{P}) = 0 \\ 1 + \dfrac{f(\mathbf{x}) - \mu(\mathcal{P})}{\alpha\,\sigma(\mathcal{P})} & , \text{ if } \sigma(\mathcal{P}) > 0 \end{cases} , \quad \forall \mathbf{x} \in \mathcal{P} \; , \qquad [2.5]$$

where $\alpha > 0$ is a free parameter that allows controlling the rate of individuals to be chosen for reproduction.

The selection is made by means of a BGA, after being tuned by the reproduction capacity r. To construct the profile of the probability density (p),two possible strategies can be envisaged:

a. eliminate the unable individuals (with $r < 0$) use only non-negative values of r as values of p;
b. translate r to obtain only non-negative values and use the result as p; more specifically:

$$p(\mathbf{x}) = r(\mathbf{x}) - \min_{\mathbf{y}\in\mathcal{P}} \{ r(\mathbf{y}) \} + \beta \; , \quad \forall \mathbf{x} \in \mathcal{P} \; , \qquad [2.6]$$

where $\beta \geq 0$ is a parameter to adjust the selection accuracy (the higher $\beta \geq 0$, the more chances for the less able individuals to reproduce).

In the first case, the selection is less accurate than in the second case, but the rate of individuals involved in the reproduction remains approximately constant.

If the exploration-exploitation trade-off is analyzed when using this selection technique, one concludes that:

- in the beginning of evolution, when the population variance ($\sigma(\mathcal{P})$) is generally large, the reproduction capacity of the fittest individuals is not excessively large (take a look again at definitions [2.4] or [2.5]), which increases the chance of other individuals to be selected for reproduction; it follows that the diversity of the population does not decrease so quickly, due to the σ-normalization;
- in the end of evolution, when the population variance is small, there is a fairly clear separation between the most suitable individuals for reproduction and the other individuals (although they too are capable of reproduction); σ-normalization ensures this property, as well.

The user can control (and even keep almost constant) the rate of individuals selected for reproduction, but, with this technique, it is still possible that the population diversity decreases too quickly in some applications.

C. Selection by Boltzmann's law

To improve the previous selection technique (through σ-normalization), a more complex strategy can be implemented. Two types of approaches are considered here:

- for the initial generations of population, it is suitable to have more *liberal (permissive)* selection rules, allowing the fertile individuals to meet less able or unable for reproduction individuals;
- as the population evolves, one notices the emergence of a sub-population of individuals with high reproduction capacity (an *elite*, in fact); now, the reproduction must become more and more *conservative*, especially for individuals of the elite; the other individuals should gradually be replaced by the elite inheritors.

The strategy is similar to the phenomenon of steel annealing, described in Section 1.5 and modeled by Boltzmann's law. For this reason, the reproduction capacity of the individuals in the population could be evaluated by means of an auxiliary parameter referred to as *temperature of selection*, $T > 0$. Like for steel, the temperature starts from quite large values and gradually decreases, across generations, following a specific program. The large temperature values are associated to a reduced capacity of reproduction, while this capacity increases with cooling.
The reproduction capacity is defined as follows, according to Boltzmann's law:

$$\boldsymbol{r}(\mathbf{x}) = \frac{\exp\left[\dfrac{f(\mathbf{x})}{T}\right]}{\displaystyle\sum_{\mathbf{y}\in\boldsymbol{\mathcal{P}}} \exp\left[\dfrac{f(\mathbf{y})}{T}\right]}, \quad \forall\, \mathbf{x} \in \boldsymbol{\mathcal{P}}\,. \tag{2.7}$$

The profile of the probability density for BGA tuning directly results from definition [2.7]:

$$\boldsymbol{p}(\mathbf{x}) = N\boldsymbol{r}(\mathbf{x}), \quad \forall\, \mathbf{x} \in \boldsymbol{\mathcal{P}}\,. \tag{2.8}$$

It is easy to show that if the temperature T increases, the difference between the individuals in terms of reproduction capacity also increases. It is up to the user to design the program of temperature decreasing, across generations. This can lead to a very fine control of the exploration-exploitation trade-off.

D. Selection by ranking

It is possible to assign each individual a rank, which generally is determined by its fitness [BAK 85]. The rank generates a certain position of the individual in the population, after ordering. Thus, one can say that some individual is privileged, common, disadvantaged (disabled), just like in case of a social group. The idea of this selection technique is then to avoid mating between individuals with similar ranks, when the population variance is high. If the variance decreases, the individuals with close ranks are more likely to combine to each other.

The rank depends on the position the individual is taking in the population. As the position is determined by the corresponding fitness, the population can be organized so that the position of each individual increases with the fitness:

$$\begin{gathered}\boldsymbol{\mathcal{P}} = \{\ \mathbf{x}_1\ ,\ \mathbf{x}_2\ ,\ \ldots\ ,\ \mathbf{x}_N\ \} \\ f(\mathbf{x}_1) \le f(\mathbf{x}_2) \le \cdots \le f(\mathbf{x}_N)\end{gathered}\,. \tag{2.9}$$

Then the rank of the individual $\mathbf{x}_n \in \boldsymbol{\mathcal{P}}$ (taking the position $n \in \overline{1,N}$ in the population) can be evaluated according to the definitions below:

$$\rho(\mathbf{x}_n) = n \qquad \text{(linear law);} \tag{2.10}$$

$$\rho(\mathbf{x}_n) = a^n \qquad \text{(exponential law, with } a > 1\text{);} \tag{2.11}$$

$$\rho(\mathbf{x}_n) = \log_a n \qquad \text{(logarithmic law, with } a > 1\text{).} \tag{2.12}$$

The rank allows computing the reproduction capacity by two methods:

a. linear:

$$\boldsymbol{r}(\mathbf{x}) = \boldsymbol{r}_{\min} + \left(\boldsymbol{r}_{\max} - \boldsymbol{r}_{\min}\right)\frac{\rho(\mathbf{x})-1}{N-1}, \quad \forall\, \mathbf{x} \in \boldsymbol{\mathcal{P}}\,, \qquad [2.13]$$

where $\boldsymbol{r}_{\min}$ and $\boldsymbol{r}_{\max}$ are parameters chosen by the user such that $\boldsymbol{r}_{\min} \le \boldsymbol{r}_{\max}$;

b. exponential:

$$\boldsymbol{r}(\mathbf{x}) = \left(\frac{\rho(\mathbf{x})}{N}\right)^{r}, \quad \forall\, \mathbf{x} \in \boldsymbol{\mathcal{P}}\,, \qquad [2.14]$$

where $r > 0$ is a parameter chosen by the user.

The denominator in definition [2.13] ($N-1$) corresponds to definition [2.10] of the rank. It can be replaced by $a^N - 1$ from definition [2.11] and by $\log_a(N) - 1$ from definition [2.12]. In order to select the parameters $\boldsymbol{r}_{\min}$ and $\boldsymbol{r}_{\max}$ in definition [2.13], the following constraint is imposed:

$$\sum_{n=1}^{N} \boldsymbol{r}\left(\mathbf{x}_n\right) = N\,, \qquad [2.15]$$

which leads to:

$$\boldsymbol{r}_{\min} + \boldsymbol{r}_{\max} = 2\,. \qquad [2.16]$$

According to Baker's idea in [BAK 85], a good enough choice, that results in an acceptable trade-off between exploration and exploitation, is the following: $\boldsymbol{r}_{\min} = 0.9$ and $\boldsymbol{r}_{\max} = 1.1$. The constraint [2.15] (or a similar one) can be imposed for the exponential and logarithmic laws of the rank.
In definition [2.14], since :

$$\frac{\rho\left(\mathbf{x}_n\right)}{N} = \frac{n}{N} \le 1\,, \quad \forall\, n \in \overline{1, N} \qquad [2.17]$$

and $\boldsymbol{r}\left(\mathbf{x}_N\right) = 1$, the parameter r plays the role of separator between able and unable individuals. The higher the r, the more visible the separator, since the reproduction capability of the individuals with small rank faster decreases towards zero. Therefore, it is recommended that this parameter first starts from small values and then increases during the evolution.

The probability profile for BGA tuning is proportional to the reproduction capability ($p(\mathbf{x}) \sim r(\mathbf{x})$, $\forall\, \mathbf{x} \in \mathcal{P}$).

By using this selection technique, the evolution could be slow, especially because a supplementary operation is added at each generation: the classification of the population like in [2.9]. In compensation, this technique has the advantage of preserving a comfortable diversity of population during its evolution (individuals with small relative fitness always are on the first positions in the population, regardless the absolute value of their fitness). The user can control quite well the exploration-exploitation trade-off by means of the provided variable parameters.

E. Selection by tournament

The previous technique presented in this section computes the fitness for all the individuals in the population. By using this technique, the computation effort could be reduced through the following strategy [GOL 91]:

1. Choose a number N_r of vacant positions in the reproduction set (to take by contest or *tournament*). Definitely, N_r is less than or equal to N. Each position has a certain importance, encoded by a parameter $\eta \in [0.5,1]$. A tournament is organized in order to fill in the vacant positions for reproduction.
2. For each offered position:
   2.1. Choose two competitors $\mathbf{x} \in \mathcal{P}$ and $\mathbf{y} \in \mathcal{P}$ at random, by means of a U-PRSG (without computing their fitness in advance).
   2.2. Compute the fitness for each competitor.
   2.3. Use a U-PRSG to generate a number $\nu \in [0,1]$.
   2.4. If $\nu \geq \eta$, the position will be taken by the best fitted competitor (with the best fitness). Otherwise, its opponent will be preferred.
   2.5. After participating at tournament, the two competitors return to the $\mathcal{P}$ population, in order to compete again.

Since $N_r \leq N$, in the worst case, it is necessary to evaluate the fitness for the entire population. But this situation is very rare. Usually, the already computed fitness values are saved, even in case an individual is competing several times. A slight improvement in terms of the computational burden is obtained by using this selection technique. However, there is an important drawback here: the user cannot control the population diversity (due to step 2.3 in the algorithm).

F. Elitist selection

By using this technique, the *elite* of the current population is transferred to the next generation. The elite population includes a number $N_e < N$ of

individuals with high fitness (eventually, the highest) that are not involved in reproduction [JON 75]. The other $N - N_e$ individuals in the population can be replaced by the inheritors obtained after applying genetic operations. Practically, the selection in view of reproduction is now made on the reduced population containing $N - N_e$ individuals who do not belong to the elite.

The exploration-exploitation trade-off is extremely sensible to the elite size N_e. Moreover, N_e has to be chosen with care for each application, since a general rule does not exist. For this reason, the elitist selection has to be combined with some other selection techniques. The combined elitist selection is very efficient in the case of noisy or non deterministic fitness, as it allows preserving the best solution, despite the intense exploration of the search space. It is wise to keep in the elite at least the best encountered individual, for it could be the global optimum itself.

2.2.2.4. *Selection for survival*

After reproduction, a group of individuals is obtained (parents, offspring, mutants). This group is too large to entirely be included in the population. As result of crossover, the group includes two parents and their two offspring. Only two individuals should then be chosen to fill in the places left vacant by the parents in population. In case of mutation or inversion, the group is formed by the couple of {initial individual, mutated individual}. Only a single individual has to be included in the population afterwards. It is thus necessary to specify how the survivors are selected. Some techniques of selection for survival are described in the sequel.

A. Generational selection

By this technique, the inheritors are always preferred, regardless their fitness or reproduction capacity. The resulting offspring of a crossover will replace the parents and the mutant resulted from a mutation or inversion will replace the initial individual. This choice is intended to maintain the diversity of the population. However, the major risk is to remove the current or the global optimum. The technique promotes exploration to detriment of exploitation.

B. Elitist selection

In this case, an elite is maintained in the population, while the other individuals are replaced with their inheritors. Although this technique can lead to a good compromise between exploration and exploitation, the diversity of population does not decrease as fast as needed and the search for the optimum can be slow.

C. Generational elitist selection

This technique is probably the best one, since the advantages of the two previous techniques are combined. If inheritors are born from reproducers, the best of reproducers and inheritors group are selected. For this reason, the

technique is known as the equivalent name *(μ+λ)-selection*. The weight is however slightly unbalanced towards exploitation, since the mutants (which are the engine of the diversity) are often removed.

2.2.3. *General structure of a GA*

Starting from the previous sections, the general structure of a GA is illustrated by the flow scheme in Figure 2.7.

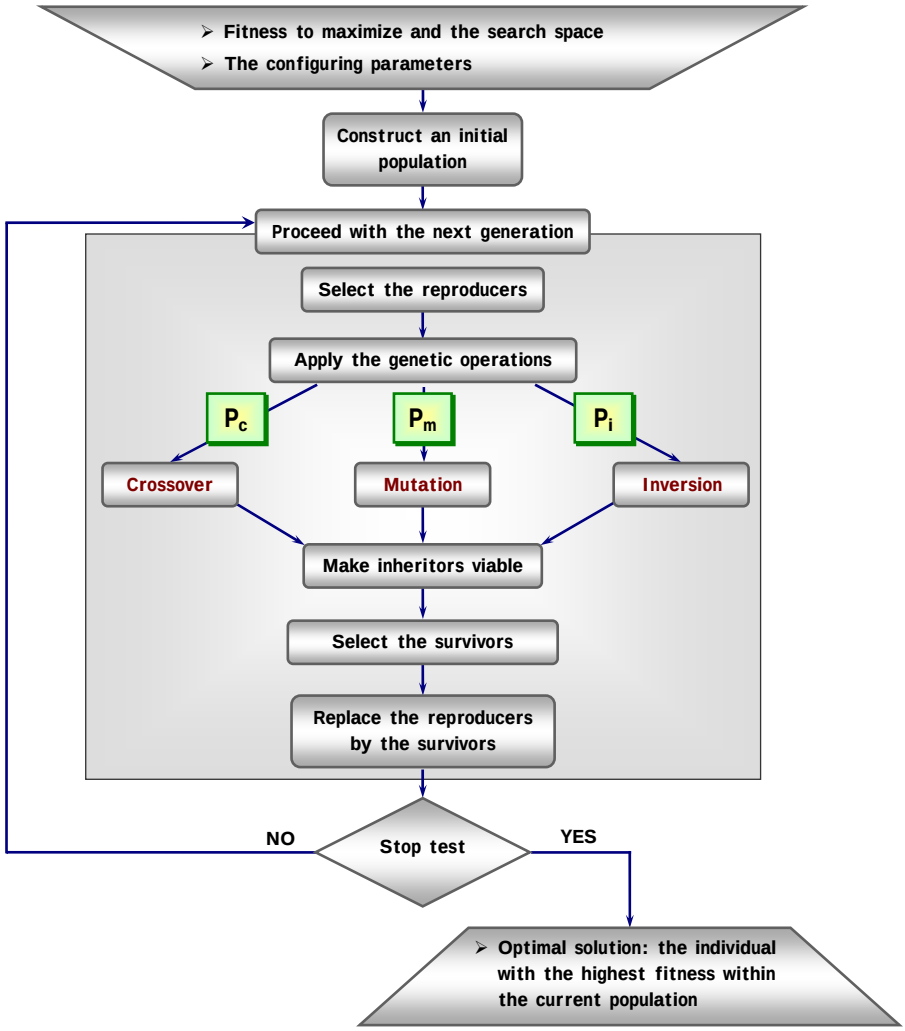


Figure 2.7. *General structure of a Genetic Algorithm.*

The success of a GA strongly depends on two factors:

1. The selection of the configuring parameters.
2. The control of the exploration-exploitation trade-off.

Also, the previous sections have suggested that, in order to configure a GA, quite a large number of parameters have to be set. In fact, one of the most important limitations of GA is caused by the lack of general rules concerning parameters tuning in the design stage. The user has to cope with rather a difficult problem, given the large number of parameters to set.

Table 2.1 shows an extensive list of configuring parameters and their usual variation ranges.

Table 2.1. *List of configuring parameters in GA design.*

- Population size: $N \in \overline{50,200}$.
- Maximum number of genetic operations to apply for a new generation: $N_{og} \in \overline{N,6N}$ (optional parameter).
- Type and structure of the crossover (pivots, length, mask, etc.).
- Crossover probability: $P_c \in [0.5, 0.95]$.
- Type and structure of the mutation (pivots, length, mask, etc.).
- Mutation probability: $P_m \in [0.005, 0.1]$ or $P_m \approx P_c$, if the crossover is not applied.
- Structure of the inversion (pivots, length).
- Inversion probability: $P_i \in [0.005, 0.1]$.
- Proportion of the elitist selection: $P_e = \frac{N_e}{N} \in [0.01, 0.3]$.
- Method for reproducers selection (and the corresponding parameters).
- Method for survivors selection.
- Control parameters for the exploration-exploitation trade-off: $\delta \in [0, 0.3]$ (if null, the user does not control this compromise).
- Stop threshold with respect to the relative value of the fitness: $\varepsilon \in [0, 0.25]$ (if null, the algorithm intends to find the global maximum of the fitness).
- Maximum number of generations during the evolution: $N_g \in \overline{50,200}$.
- Survival factor: $S \in \overline{2,10}$ (higher for the initial population and the first 5-10 generations).

It is not mandatory to use all those parameters in a GA, but it is suitable to select the most appropriate parameters for each application. To get the "right" parameters, the GA usually runs several times, each time with a different starting configuration. To shorten this learning time, the user should know very well in advance the nature

and, if possible, the shape of the fitness to be maximized. For this reason, many scientists consider that obtaining a good GA is pure art.

The control parameter for the exploration-exploitation compromise, δ, is used in the most complex GA, where the searching time is less important than the requirement to find the global optimum. The variance of each new population is compared to the variance of the previous population. If the population variance has a relative increase greater than δ, then the population tends to become too diverse and the exploration is favored to detriment of exploitation. In this case, it is suitable to increase the proportion of elitist crossovers (between the most fertile individuals). If, on the contrary, the relative variance decrease is greater than δ, then the population tends to become too uniform, and probably will be dominated by the elite. In this case, the diversity is regained by mutations, inversions and crossovers between the individuals with very different values of fitness.

To build the initial population required by the diagram of Figure 2.7, usually, uniform sampling of the search space is performed (or of an area of this space that presumably contains the desired optimum).

Several tests are available to stop the evolution of the population towards an optimum. The parameters associated to such tests are specified in Table 2.1, at the end of the list.

Normally, it is required that the optimum is determined with a certain precision. More specifically, the search ends when:

$$\frac{f\left(\mathbf{x}_N\right) - f\left(\mathbf{x}_{N-1}\right)}{f\left(\mathbf{x}_{N-1}\right)} < \varepsilon, \qquad [2.18]$$

where $\mathbf{x}_{N-1}$ and $\mathbf{x}_N$ are the best individuals of the current population, after ordering it in ascending order of the fitness (see [2.9]). If $\varepsilon = 0$, this test should be combined with other tests, since there is the risk to unnecessarily lengthen the search.

A stop test often used in the applications results from the necessity of limiting the population evolution to a number of generations, say N_g. This test is very useful in case of fitness with many maxima with close values (which could make the GA to oscillate).

Finally, a stop test that seems suitable for all evolutionary algorithms concerns the ability of the best individual in the population to survive across generations. If this individual is not replaced by a better one, during S successive generations, then one can assume that the wanted optimal point has been found. In the beginning of evolution, the population is rather located in the exploration phase and a false optimum can survive long enough. In this case, a large value should be chosen for S, in order to avoid stopping the search prematurely. Later on, after obtaining a good balance between exploration and exploitation within the GA, S can be decreased.

Normally, stopping the search by using the survival factor is better fitted to the evolutionary algorithms than stopping it by accuracy requirement (condition [2.18]), which is quite subjective. This test is considered by many scientists the main stopping instrument in the evolutionary programming. But a "good" GA (i.e. rapidly convergent) uses a combination of stop tests and not a single test.

Behind the diagram in Figure 2.7, many implementation details that can help the designer to increase the effectiveness of a GA are hidden. For each application, it is recommended to find the best GA adaptation. An example of GA implementation is presented in the end of this section.

2.2.4. *On the convergence of GA*

Analyses of GA convergence were developed by many scientists. For GA and any other evolutionary process, it is not possible to prove general and sound convergence results. All we can do is to make some assumptions about the convergence. In other words, the convergence of GA is not a provable phenomenon, but observable only. One noticed that, in most applications, the GA converged to an optimum solution of the fitness. But this remark could not be generalized to formulate a conclusive convergence result. For this reason mainly, the GA could inexplicably fail in certain applications, despite their utility that seems to be so general.

John Holland, the father of GA, developed in his publications (in principal [HOL 75] and [HOL 92]) an analysis of convergence that was focused on parts of optimal chromosomes referred to as (*constructive*) *schemes*. A *scheme* in a chromosome is like a gene delimited by two well known side symbols, but with unknown inner symbols. In fact, a scheme is described as follows:

$$\cdots \boxed{s_b \mid * \mid * \mid \cdots \mid * \mid * \mid s_e} \cdots , \qquad [2.19]$$

where s_b and s_e are the symbols in the beginning and in the end of each scheme, respectively. The two symbols are well known in advance. In definition [2.19], on the star positions all accepted symbols could appear. Note that the scheme often represents a part of the chromosome and very rarely the whole chromosome. For example, a binary scheme is: $1{*}{*}{*}{*}0$, while a scheme corresponding to the traveling salesman problem is $C{*}{*}{*}{*}H$. Practically, a scheme is some ad hoc gene of chromosome, associated to a set of alleles that starts with the symbol s_b and ends with the symbol s_e. An instance of a scheme is called *realization*. For example, a realization of the binary scheme $1{*}{*}{*}{*}0$ is 101110. Similarly, the chromosomes of an evolving population are *realizations* of a stochastic process that governs this evolution.

By definition, the *order* of a scheme is the number of well-specified symbols. Thus, the general scheme [2.19] is of second order, while the schema $10{*}{*}{*}0$ is of

third order. The length of a scheme is also an important parameter, which measures the distance between the symbols of its sides. For example, the length of scheme $1{*}{*}{*}0$ is 6.

The schemes play an important role in the convergence of the GA. To demonstrate this role, Holland firstly formulated the hypothesis below (very similar to Ergodic Hypothesis in Physics):

- The implicit parallelism hypothesis (IPH)
  The average fitness estimated from a finite population of chromosomes at a certain instant of the evolution represents an approximation of the instantaneous mathematical expectation of the fitness of all chromosomes that are involved in this evolution (even if they do not belong to the population).

This hypothesis suggests that, during the evolution of a GA, several solutions can be tested in parallel, within different populations, in order to solve the optimization problem. Therefore, all GA have a degree of parallelism that favors finding the global optimum.

Starting from the IPH, one can observe that the schemes leading to low average fitness have many realizations in the population, during its evolution. This involves such schemes are more tested than the others. However, the schemes of the fitness with values over the average and, in addition, that do not abruptly change the fitness after genetic operations, are included in the possible solutions. In other words, the possible solutions contain schemes that can survive in combination with another ones.

To better understand this phenomenon, consider an example. The maximum of fitness:

$$f(x) = -x^2 + 2\pi x - 1, \quad \forall\, x \in \mathbb{R}, \qquad [2.20]$$

is $\left(x_{\max} = \pi \cong 3.141592,\ f_{\max} = \pi^2 - 1\right)$. Suppose that the maximum point has to be found with $\varepsilon = 10^{-6}$ accuracy by a GA designed on this purpose. The chromosome consists of seven decimal digits of the optimal point (one integer and 6 decimals):

e	d_1	d_2	d_3	d_4	d_5	d_6

. Crossover and inversion do not change the basic definitions, but for the mutation, a new rule is stated: the mutant corresponding to digit $c \in \overline{0,9}$ is the number $9 - c$. During the evolution, one notices, for example, that there are more realizations of the $0{*}{*}{*}5$, $1{*}{*}{*}5$ and $2{*}{*}{*}5$ schemes (at the beginning of the chromosome), which lead to fitness values below the average of the population, than the realizations of the $3{*}{*}{*}5$ scheme (with higher values of fitness). At the same time, the survival factor of the scheme $3141{*}{*}2$ is sufficiently large, since this scheme is, in fact, highly resistant to genetic operators.

In general, it is interesting to estimate how many realizations of a scheme can have in a population, on average, during its evolution. In order to compute this, it is useful to formalize the framework.

Let $p \in \mathbb{N}$ be the index of the current generation of the population. Then, the corresponding population is $\mathcal{P}_p$. One can estimate the number of inheritors of a chromosome $\mathbf{x} \in \mathcal{S}$ in the population $\mathcal{P}_p$, as follows:

$$N_p(\mathbf{x}) = \frac{f(\mathbf{x})}{\mu(\mathcal{P}_p)}, \qquad [2.21]$$

where $\mu(\mathcal{P}_p)$ stands for the average fitness over the population. If the chromosome fitness is higher than the average, then it is very likely that one of its inheritors will be included into the population. Otherwise, the inheritor seemingly is not entering the population (or it will be removed from the population in a future generation, if already included).

Let $\mathbf{s}$ be an arbitrarily chosen scheme. The chromosomes of the $\mathcal{P}_p$ population could contain this scheme or not. Denote by $\mathcal{P}_p(\mathbf{s})$ the group of $\mathcal{P}_p$ chromosomes that contain the realizations of the $\mathbf{s}$ scheme. If $\mathcal{P}_p(\mathbf{s})$ is not empty, then denote by $N_p(\mathbf{s}) \geq 1$ the number of its chromosomes. The average fitness of pattern $\mathbf{s}$ in the population $\mathcal{P}_p$ can then be estimated as follows:

$$\mu_p(\mathbf{s}) = \frac{1}{N_p(\mathbf{s})} \sum_{\mathbf{x} \in \mathcal{P}_p(\mathbf{s})} f(\mathbf{x}) . \qquad [2.22]$$

In this context, the following result holds, as proven in [HOL 75]:

Holland's theorem of schemes

A statistical estimation of the number of chromosomes that contain the realizations of the scheme $\mathbf{s}$ in the next population $\mathcal{P}_{p+1}$, namely $N_{p+1}(\mathbf{s})$, is :

$$\mathrm{E}\{N_{p+1}(\mathbf{s})\} \geq N_p(\mathbf{s}) \frac{\mu_p(\mathbf{s})}{\mu(\mathcal{P}_p)} \left[1 - P_c \frac{\lambda(\mathbf{s}) - 1}{L - 1}\right] (1 - P_m)^{\nu(\mathbf{s})}, \qquad [2.23]$$

where : $\lambda(\mathbf{s})$ is the length of scheme $\mathbf{s}$;

$\nu(\mathbf{s})$ is the order of scheme $\mathbf{s}$;

L is the chromosome length.

This result has some implications concerning the evolution of chromosomes population. Firstly, the lower limit of the mathematical expectation expressed in [2.23] shows that the schemes with small values in length and order are more likely to survive in the subsequent generations than other ones (observe the last two factors in the expression). This is quite natural, since in the vicinity of the optimum, the most symbols of the optimal chromosomes are already specified and only a small number of them has to be determined. In the previous example, the chromosomes $31415*2$ containing the realizations of the scheme $5*2$ as the last gene, have a high capacity of survival. This scheme is of length 3 and order 2. Indeed, the chromosomes $31415**$ form an elite population. For them, the lower bound of Holland's theorem takes high values.

The first two factors expressing the lower limit lead to:

$$N_p(\mathbf{s})\frac{\mu_p(\mathbf{s})}{\mu\left(\mathcal{P}_p\right)} = N\frac{\sum_{\mathbf{x}\in\mathcal{P}_p(\mathbf{s})} f(\mathbf{x})}{\sum_{\mathbf{x}\in\mathcal{P}_p} f(\mathbf{x})}, \qquad [2.24]$$

which shows that the high fitness schemes have more chances to survive than other schemes.

A direct consequence of Holland's theorem highlights the perpetuation of the schemes across the generations. In general, during the evolution of a GA, the number of the realizations of short schemas with reduced order, but leading to higher values of fitness, increases. These realizations can serve as constructive blocks to exchange by crossovers, in order to obtain optimal chromosomes that contain schemes of larger order. The only problem is to identify these constructive blocks, so that the crossovers can be applied successfully.

Another interesting observation related to schemes was formulated in [GRE 93]. By difference from the above property, which describes the normal functioning GA, in this publication, here some GA limitations are highlighted. In some cases, the group of short length schemes of reduced order, but having a high fitness, constitutes a trap for the GA. The population can finally fall into this trap, since those schemes associate together into a pretty strongly dominant group. Therefore, a visible sub-optimal solution is found or, more often, the GA becomes oscillating within a few schemes group that cannot be abandoned.

The study of GA convergence has advanced beyond Holland's results.

An ideal AG, based on a sound formalization, known as *Vose-Liepins model*, was proposed in [VOS 91]. The basic assumption in this algorithm is that the population has an infinite number of chromosomes and one could know in advance the scheme leading to the global optimum (*the optimum scheme*). For this reason, the fitness is never computed during the evolution. The goal here is to determine the chromosome that includes all the optimal schemes, which is equivalent to maximizing the fitness. In fact, the Vose-Liepins GA version with infinite

population is a procedure that simulates the evolution of biological systems, as governed by natural selection. Although this approach leads to an ideal algorithm, impossible to implement, it reveals a manner of evolution quite close to reality. Moreover, it explains why a GA is rather convergent than divergent, its convergence limit being in fact a fixed point that attracts the infinite population of chromosomes. In the vicinity of fixed point, the selection rate in the population is approximately constant for each chromosome. This leads to the stabilization of the population that does not evolve any more and thus allows discovering the optimum.

Another type of GA is based on the Nix-Vose model of finite populations [NIX 91]. This model is related to the Markov Chains Theory. Any population with finite size, say $N \geq 1$, evolves along a Markov chain, since its current structure is determined by the structures across the past generations. All finite populations (of size N) stand for the states of the stochastic process associated to the evolution by natural selection. Each initial population follows a trajectory of Markov type, passing from one state of the stochastic process to another. By using the mathematical formalism of Markov chains, the following properties of the related GA are shown:

a. If $N \to \infty$, the finite population model begins to behave as a model with infinite populations (although the optimal schemes are unknown).
b. If $N \to \infty$, there are some trajectories that converge to all the *stable points* of stochastic process, which, in fact, are located in the fitness optima.

To implement the GA with finite populations, the main problem is to find one or more trajectories towards the defined stable points. Since there is no rule regarding the selection of the starting population, the convergence of the GA with finite populations can not be guaranteed. However, Vose formulated quite a plausible assumption that serves as base for constructing a GA with high probability of convergence:

Vose's conjecture

If $N \to \infty$, the time spent by a GA with finite population in the vicinity of the unstable defined points (false optimum, i.e., for example, inflection points with zero derivative) tends to zero.

The conjecture asserts that, if the initial population was not properly selected and its size is large enough, the GA will end quickly, without finding an optimum. Naturally, the GA must run again, several times, with different initial populations, in order to find a trajectory towards one of the stable fixed points. This confirms the practical observation that the main problem in the design of a GA is the calibration of the configuring parameters, including the initial population. As previously mentioned, this calibration is only possible by repeated attempts, specific to each application, made until the best combination of the parameters is obtained.

For further mathematical details of the formal models of GA, the reader is recommended to consult, for example, [MIT 95].

2.2.5. *How to implement a genetic algorithm*

In order to have the ability to design a GA, it is necessary to answer the following questions:

- How to define the chromosome?
- How to make the chromosomes viable?
- How to generate/select the initial population?
- What are the configuring successful settings?
- Which is the best stop test?

Other technical issues arise during the implementation. According to most scientists' opinion, the "good" GAs are among the most sophisticated and difficult to implement metaheuristic optimization procedures.

Instead of giving general rules on the GA implementation, an example of using a GA in a specific application is given. The application led to quite a difficult granular optimization problem, where the fitness is strongly irregular.

The application deals with the detection of mechanical defects of the bearings, using the vibration signals, acquired by means of an accelerometer. Usually, these signals encode information about 12 possible single faults and also about a large number of multiple defects [STE 03a]. From the beginning, when these defects appear, the vibration starts to exhibit abnormal variations. The problem is to decode the information about the abnormal operation, in order to prevent the mechanical damage of the system that includes the defect bearing. This application was in-depth detailed in [STE 03a] and [STE 03b]. In this section, the discussion is focused on the design of the GA, considered as the optimization tool.

Since the mechanical vibrations are non-stationary signals (with time variant spectrum), a method to solve the problem is to build and operate with non-stationary models associated to this type of signals. An extremely interesting model can be determined from a dictionary of waveforms, where each "word" is a wavelet [DAU 92], [STE 10]. In fact, the dictionary is of *time-frequency-scale* type [COH 95], where the characteristic parameters of its wavelets are: the scaling factor σ_0, the time shifting τ_0 and the harmonic modulation with pulsation ω_0. The base wavelet that can generate the entire dictionary should be adapted to the nature of the vibration signal. For example, the Morlet-Gabor wavelet, below:

$$g(t) = \frac{1}{\sqrt[4]{\pi\sigma^2}} e^{-\frac{(t-t_0)^2}{2\sigma^2}}, \quad \forall t \in \mathbb{R}, \qquad [2.25]$$

with $\sigma > 0$ as bell aperture and $t_0 \in \mathbb{R}$ as central instant, is often employed in order to construct time-frequency dictionaries in applications.

The wavelets of the dictionary, also called *atoms*, are generated as follows (given that the vibrations are discrete signals):

$$g_{[m,n,k]}[l] = \sigma_0^{-m/2} \exp\left(-j\,\omega_0 \sigma_0^{-m} k l T_s\right) g\left(\sigma_0^{-m}\left(l T_s - n\tau_0\right)\right), \quad \forall\, l \in \mathbb{Z}, \qquad [2.26]$$

where: $j^2 = -1$, T_s is the sampling period, $m \in \overline{0, M_{\max}}$ is the scaling index, $n \in \overline{N_{m,\min}, N_{m,\max}}$ represents the index of the time interval, and $k \in \overline{0, K_{m,\max}}$ is the index of frequency modulation. Figure 2.8 shows nine atoms of the dictionary generated by the wavelet [2.25].

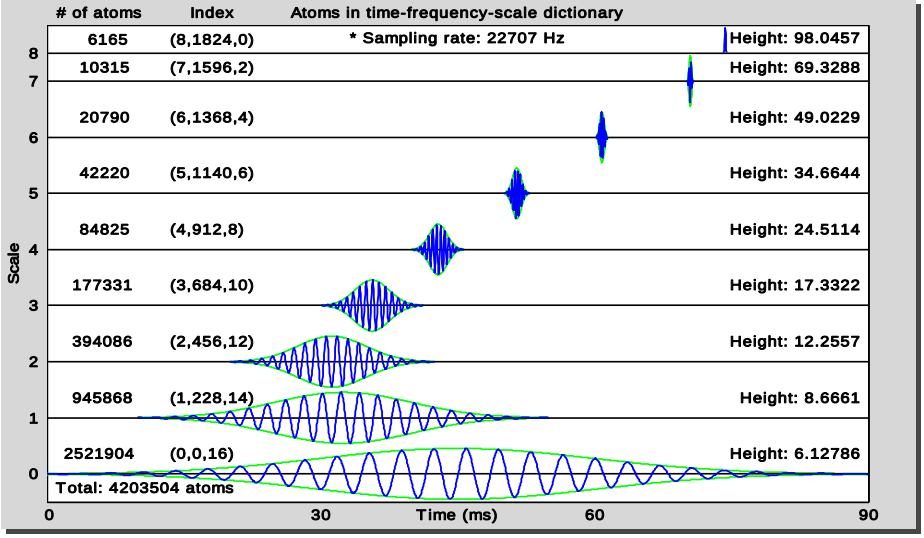


Figure 2.8. *Examples of time-frequency-scale atoms on a waveforms dictionary.*

One can thus easily notice the effect of the three operators used for generating the atoms: scaling, temporal shifting and frequency modulation. In this example, the dictionary, denoted by $\mathcal{D}[g]$, includes more than 4 million atoms, organized on 9 representation scales. The limits of the index variation can be computed from the vibration signal v (which contains the information about the possible defects). The maximum number of representing scales ($M_{\max}$) and the maximum number of frequency bands on a scale ($K_{m,\max}$) are determined from the bandwidth of signal v (after some preliminary filtering was applied). The limits of the time shifting for a scale ($N_{m,\min}$ and $N_{m,\max}$) depend on the width of the wavelet support for the m scale and on the number of acquired vibration samples (v), denoted by N_v. The central moment of the basic wavelet (t_0) is naturally chosen in the middle of the vibration: $\left(N_v - 1\right)T_s/2$. This means: $\sigma = t_0/3$, due to Gaussian function properties. (The practical support of the basic wavelet is identical to the vibration support). Additionally, the parameters of the three operators are set as follows:

$\sigma_0 = 1/2$, $\tau_0 = T_s$ and $\omega_0 = 2\sqrt{\ln 2}/\sigma$ (see [STE 03a] for more details). This leads to a number of 4 203 504 generated atoms.

The $\mathcal{D}[g]$ dictionary is used to "translate the phrase" represented by the vibration v. This translation involves performing the projection of the vibration on the space spanned by the dictionary, as suggested in Figure 2.9. It is unlikely that the vibration is a part of the dictionary generated space. As in any other dictionary, in $\mathcal{D}[g]$ some "words" are missing, which makes them untranslatable. The difference between the vibration v and its projection v_D on the spanned space is seen like a noise Δv, produced by all the untranslatable "words" that can be found in the phrase, but not in the dictionary.

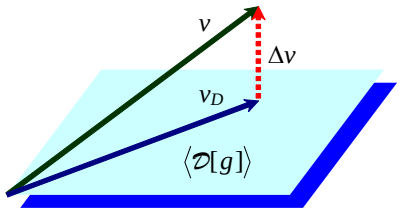


Figure 2.9. *Noise attenuation of a signal by using a waveform dictionary.*

The v_D projection actually encodes the information related to bearing defects. Note however that this manner of noise attenuation is specific to the selected dictionary. Changing the dictionary (i.e., changing the basic wavelet) means in fact changing the noise.

Now, the problem is reduced to the construction of projection v_D. In [MAL 93], a simple and ingenious method was proposed: *the matching pursuit of the best atoms*. When projecting the vibration on the dictionary space, one seeks firstly *the best suited atom that matches the vibration*. This property is quantified by the magnitude of the scalar product between each atom and the vibration. This magnitude has to be maximum for each iteration. Once the most suitable atom is found, it is removed from the vibration. The remaining residual stands now for a new vibration that requires a new translation with the help of dictionary. It is sufficient to find the most suitable atom for this residual too. Once the new atom is found, it is removed from the residual as well, thus producing a new residual. This iterative pursuit process continues until a sufficiently long sequence of best matching atoms is obtained. Since the energy of the current residual is smaller than the energy of the previous residual, the iterative process stops when the energy decreases below a threshold selected by the user. The remaining residual is then the noise Δv, while the linear combination of the best matching atoms defines the projection v_D.

The general expression of the iterative pursuit process is written as follows:

$$\Delta^{q+1}x \equiv \Delta^q x - \left\langle \Delta^q x, g_{\lfloor m_q, n_q, k_q \rfloor} \right\rangle g_{\lfloor m_q, n_q, k_q \rfloor}, \quad \forall\, q \geq 0, \qquad [2.27]$$

where $\Delta^q x$ and $\Delta^{q+1} x$ are successive residuals (by convention, $\Delta^0 x \equiv v$) and $g_{\lceil m_q, n_q, k_q \rceil}$ is the current best matching atom found into the dictionary $\mathcal{D}[g]$. Since the energy is conserved in expression [2.27], i.e.:

$$\left\|\Delta^q x\right\|^2 \equiv \left|\left\langle \Delta^q x, g_{\lceil m_q, n_q, k_q \rceil} \right\rangle\right|^2 + \left\|\Delta^{q+1} x\right\|^2, \quad \forall\, q \geq 0 \qquad [2.28]$$

(even though the best matching atoms are not necessarily orthonormal), it results that the stop test has the following form :

$$\left\|\Delta^{q+1} x\right\|^2 < \varepsilon, \qquad [2.29]$$

for a threshold $\varepsilon > 0$ a priori set, which leads to a maximum number of iterations, say $Q \in \mathbb{N}^*$. The obtained projection is then:

$$x_D \equiv \sum_{q=0}^{Q} \left\langle \Delta^q x, g_{\lceil m_q, n_q, k_q \rceil} \right\rangle g_{\lceil m_q, n_q, k_q \rceil}, \quad \forall\, q \geq 0\,. \qquad [2.30]$$

Now, the coefficients $\left\{\left\langle \Delta^q x, g_{\lceil m_q, n_q, k_q \rceil} \right\rangle\right\}_{q \in \overline{0,Q}}$ directly decode the information containing the possible faults. The defects lie in relatively narrow frequency bands, of specific scales and cause micro-shocks at random instants in the harmonic behavior of the vibration. The atoms of the dictionary can, therefore, isolate these defects.

The major drawback of this method, as it easily can be noticed, is related to the searching duration of the best matching atom for each residual. Testing all the atoms in the dictionary, for each iteration, is a naive (and inefficient) approach, given their number. The exhaustive search is thus excluded, at least for small values of the scale index. (Figure 2.8 shows that, in this example, there are over 2.5 million atoms for the null index scale and approximately 6000 for the maximum index scale, equal to 8.) Even if one agrees to perform an exhaustive search on the maximum index scale, this approach is not effective for the other scales.

Finding the most suitable atom represents in fact solving the following granular optimization problem:

$$\max_{m,n,k} \left| \sum_{l \in \mathbb{Z}} \Delta^q x[l] \overline{g_{\lceil m,n,k \rceil}[l]} \right|, \quad \forall\, q \geq 0\,, \qquad [2.31]$$

where $\overline{a}$ is the complex conjugate of the number $a \in \mathbb{C}$. Note that the problem [2.31] is supposed to be solved for each residual $\Delta^q x$. The current fitness is:

$$f_q[m,n,k]=\left|\sum_{l\in\mathbb{Z}}\Delta^q x[l]\overline{g_{[m,n,k]}[l]}\right|,\quad \begin{bmatrix}\forall m\in\overline{0,M_{\max}}\\ \forall n\in\overline{N_{m,\min},N_{m,\max}},\ \forall q\geq 0.\\ \forall k\in\overline{0,K_{m,\max}}\end{bmatrix} \qquad [2.32]$$

The exhaustive search performed only on the highest scale leads to fitness variations illustrated in Figure 2.10, for the first 4 residuals.

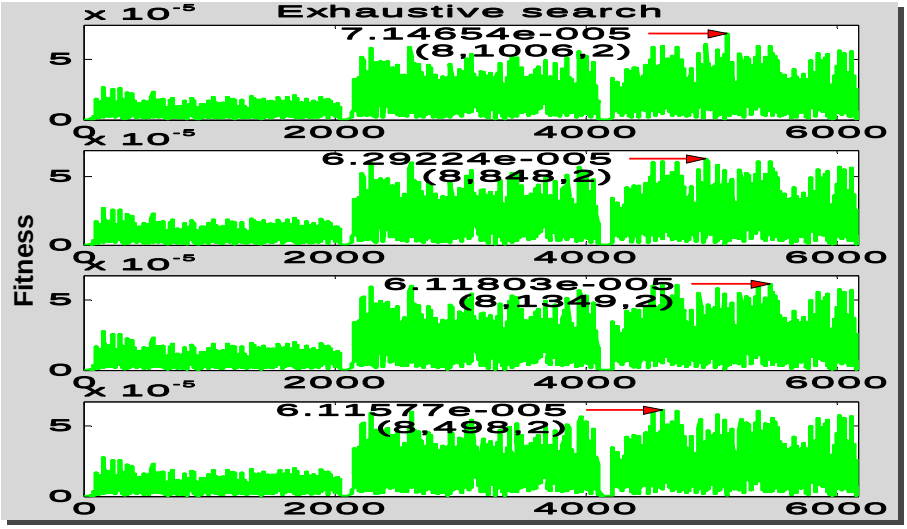


Figure 2.10. *Illustration of the strong fractal nature of the fitness in the problem of bearings mechanical faults detecting.*

The irregular (fractal) nature of this criterion, as well as the large number of atoms to be tested for other scales, naturally led to idea of using a metaheuristic to solve the problem. A GA can thus be used in this aim.

In the design of the GA, one starts from the definition of chromosomes. Naturally, the 3 binary representations of the parameters determining the atoms, meaning the indexes m, n and k could constitute the genes of the generic chromosome γ, as suggested in Figure 2.11.

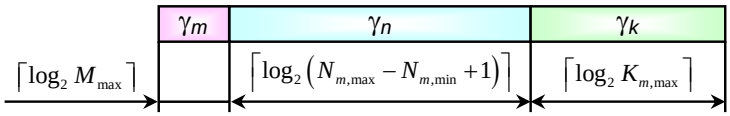


Figure 2.11. *Possible definition of the chromosome in the problem of bearings mechanical faults detection.*

The γ_n gene length is constant across the scales (since $N_{m,\max} = N_v - 1$ and $N_{m,\min} = 1 - N_v$), but the length of the γ_k gene decreases with the scale index m (according to the Uncertainty Principle [STE 10]).

However, since the number of dictionary scales is substantially smaller than the number of time intervals and the number of frequency bands, a smaller chromosome, $\gamma_{n,k}$, composed by the last two genes, can be employed. Also, the GA can evolve with parallel populations, one for each scale. After finding the optimal chromosome for each population, the best of them is selected, by exhaustive search. In the example of Figure 2.8, this strategy results in the final comparison of fitness for 9 atoms.

To determine the population to evolve, the three genetic operations are applied. (One should always make the inheritors viable before employment.) The probabilities of the genetic operations decrease: $P_c >> P_m \geq P_i$ (the crossover probability being substantially larger than the other two probabilities). After performing some, one concluded that, for this application, the best types of crossovers and mutations involve *stochastic masks*. This means pseudo-randomly chosen pivots depending on the probabilities P_c, P_m and P_i are generated. For the inversion, this operation is applied to the bits between two consecutive pivots (excluding the pivots), whereas if the last pivot is missing, the inversion area is set up to the chromosome LSB (which is excluded as well). For example, the chromosome 10 0111 1101 1011 and the mask 01 0000 1000 1000, produce by inversion the mutant 10 1110 1011 1101. (The last pivot is automatically set to the LSB.) To make inheritors viable, since the genes are binary and the searching space limits are employed for each scale, it is only sufficiently to use the modulo operation.

How to use genetic operations and the associated probabilities? Consider a binary generic chromosome. Normally, each bit of the chromosome has the same chance of being selected as a pivot in one of associated genetic masks. However, each bit of such a mask has a different probability, depending on the used genetic operation. Since three masks of the same length are employed, all the bits (from MSB to LSB) are associated to six values grouped in pairs: $\{b_{c,0}, b_{c,1}\}$ for the crossover, $\{b_{m,0}, b_{m,1}\}$ for the mutation and $\{b_{i,0}, b_{i,1}\}$ for the inversion. The probability profile of the 6 values is:

$$\wp = \begin{bmatrix} 1-P_c & P_c & 1-P_m & P_m & 1-P_i & P_i \\ b_{c,0} & b_{c,1} & b_{m,0} & b_{m,1} & b_{i,0} & b_{i,1} \end{bmatrix}, \quad [2.33]$$

with the sum equal to 3. The BGA can be used to generate a selection set of length $M = \left\lceil \frac{3}{\min \wp} \right\rceil$ (the roulette resolution, see Appendix B). It follows that the

selection set thus generated contains at least once each of the six values. For each position, at least 3 stochastic selections are performed for the selection set, by using a U-PRSG. The goal is to specify the bit values corresponding to each mask. If, for example, the bit of the crossover mask has already been selected ($b_{c,0}$ or $b_{c,1}$), the roulette gambling continues until one of the bits of mutation mask or inversion mask appears. The gamble stops when all 3 bits have been selected. Obviously, $b_{c,0} = b_{m,0} = b_{i,0} = 0$ and $b_{c,1} = b_{m,1} = b_{i,1} = 1$, although they have different probabilities, depending on the mask they belong to.

The genetic operations are separately applied for each of the chromosome genes $\gamma_{n,k}$, because of their different natures. After applying the crossover between two chromosomes, 4 offspring genes result (2 of type γ_n and 2 of type γ_k). Consequently, 2 parents and 4 offspring chromosomes are competing for the two vacant places in the population. By mutation or inversion, one mutant results from each chromosome.

The next issue concerns the selection of the chromosomes in view of reproduction. To pass to a new generation, one can adopt an elitist generational strategy. The elite size, N_e, is quite small (for example, 10% of the population size). Usually, the elite seldom changes, but updates are applied to each generation, just in case. The other chromosomes $N - N_e$ must be chosen by genetic operations applied to existing chromosomes. Note that each population should only contain different individuals, in order to make the search more effective. The elitist generational strategy is summarized in Figure 2.12.

In order to evolve from population $\mathcal{P}_p$ to next population $\mathcal{P}_{p+1}$, a transient (ephemeral) population $\mathcal{R}_p$ is used. In fact, $\mathcal{R}_p$ is a mating pool that includes all suitable chromosomes for reproduction. The number of places in $\mathcal{R}_p$ equals the number of places to fill in $\mathcal{P}_{p+1}$ (i.e. $N - N_e$). The vacant places are taken as follows. In case of crossover, the best 2 chromosomes of 2 parents and 4 offspring are selected. In case of mutation and inversion, the best chromosome of the pair {initial chromosome, mutant chromosome} is selected. To preserve some

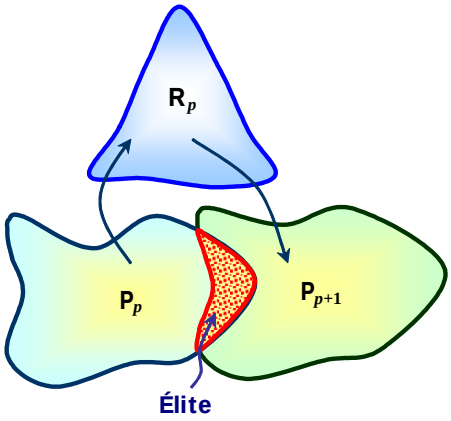


Figure 2.12. *Passing to the next generation by using the elitist generational strategy.*

diversity in population $\mathcal{P}_{p+1}$, a small number of places (usually less than 5%) are taken by randomly generated chromosomes (after being made viable), regardless their fitness. A simple and effective technique is used in this aim: to complete the population $\mathcal{P}_{p+1}$, set first a maximum number of genetic operations N_{og} to apply. Then, if after applying the N_{og} genetic operations the population $\mathcal{P}_{p+1}$ is incomplete, the remaining places are taken by randomly selected chromosomes.

How to build the transient population? Unlike the populations $\mathcal{P}_p$ and $\mathcal{P}_{p+1}$ (that have to include different chromosomes), the places in population $\mathcal{R}_p$ are taken by *the most representative* chromosomes. The representativity of a chromosome has to reflect its reproduction capacity and can be computed by using one of the strategies in paragraph 2.2.2.3. In this GA, a strategy inspired from Boltzmann's law was used. The representativity is estimated according to the definition [2.7]. The peculiarity of this strategy is illustrated by the computation method for the annealing temperature, T, which is adaptively performed across generations, as follows:

$$T[p] = N \frac{\max_{\gamma \in \mathcal{P}_p} \{ f(\gamma) \}}{\sum_{\gamma \in \mathcal{P}_p} f(\gamma)} = \frac{\max_{\gamma \in \mathcal{P}_p} \{ f(\gamma) \}}{\mu(\mathcal{P}_p)}, \quad \forall\, p \geq 0 . \qquad [2.34]$$

This definition [2.34] is based on the observation that the population variance (which is the engine of adaptation) is correlated to the ratio between the best fitness of the current population and its average. The higher this ratio, the more dispersed the population. Thus the temperature acts like in case of thermal annealing. In Figure 2.13, the variation of this temperature across the generations of populations on 8 scales is shown. (On the last scale of the dictionary, that contains few atoms, an exhaustive search is performed though.) The shape of temperature variations is an indirect proof of GA convergence, that becomes faster and faster as the scale index increases (as expected).

To select the reproducers, the probability profile for the corresponding BGA matches the representativity. (The elite is excepted from selection.) Some chromosomes of the population (the most representative ones) are included in population $\mathcal{R}_p$, with several clones, whereas other chromosomes (with small representation) could miss.

In GA design, another issue is the following: how to generate the initial population? Moreover, given the specific optimization problem [2.31], an initial population is required for each new vibration residual. From the beginning, one can remark that the GA are not very sensitive to the initialization, which constitutes a significant advantage. For the $\mathcal{D}[g]$ dictionary, the initial population corresponding to start the search for the best matching first atom is constructed by using the uniform padding technique. Figure 2.14 shows the essential of this technique.

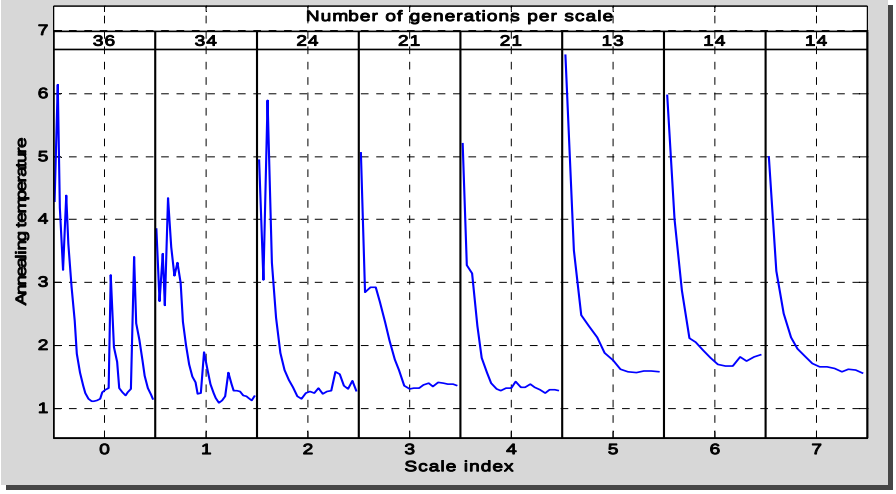


Figure 2.13. *Adaptive variation of the annealing temperature in the problem of bearings mechanical faults detecting.*

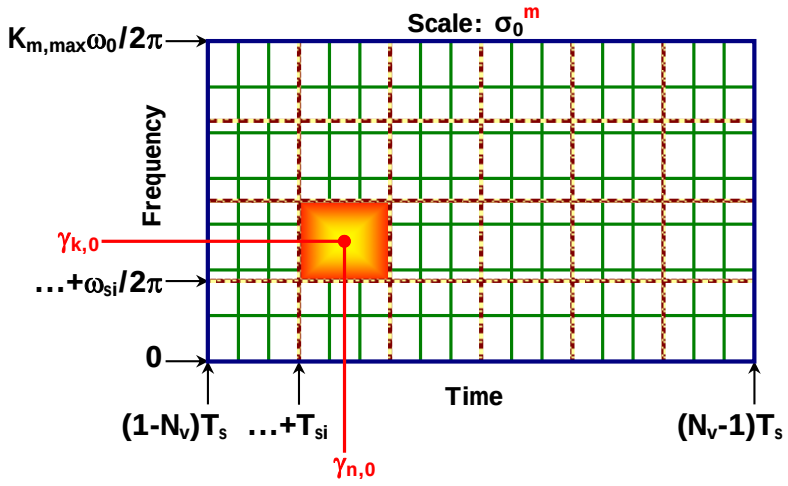


Figure 2.14. *Generate the initial population in the problem of bearings mechanical faults detecting.*

For each scale index $m \in \overline{0, M_{\max}}$, the time-frequency space has to be sampled with initial period T_{si} and initial frequency $\omega_{si}/2\pi$. The space is bounded by the time instants $\{(1-N_v)T_s, (N_v-1)T_s\}$ and the frequency instants $\left\{0, \frac{K_{\max}\omega_0}{2\pi}\right\}$. The

parameters T_{si} and ω_{si} are selected such that the total number of array cells is N (the population size). Thus:

$$T_{si} = \frac{2(N_v - 1)}{\sqrt{N}} T_s \quad \text{and} \quad \omega_{si} = \frac{K_{\max}}{\sqrt{N}} \omega_0 . \qquad [2.35]$$

The chromosomes are selected as close as possible to the middle of each cell (recall that genes are binary representations of integers). This population leads to a large diversity, as required for "good" initializations of GA.

For the next residual, one can exploit an interesting property of the fitness, as shown in Figure 2.10. As a remark, for higher scale indices, the fitness profile does not significantly change from one residual to another, since the removed atom only affects quite a small part of the vibration. (Also, many projections, once being computed, are preserved for the next residual, which optimizes the search.) Consequently, for larger scales, one can start from the last population of the current residual as initial population of the next residual. For smaller scales, it is better to start from a uniform initial population.

In the genetic procedure, several stop tests are applied. Algorithm 2.1 includes all these tests and answers to the remaining issues, concerning the design of the AG.

Algorithm 2.1. *Genetic procedure for solving the matching pursuit problem in a time-frequency-scale dictionary.*

1. Input data :

- Acquired vibration signal v (of size N_v; for example, $N_v = 2048$ samples).
- Sampling period, T_s (usually set after a primary filtering of the raw vibration, with a known bandwidth; for example, $T_s = 0.039$ ms).
- GA parameters:
  - population size, $N = 100$;
  - maximum number of genetic operations to apply, in order to proceed to the new generation: $N_{go} = 450$;
  - type and structure of the genetic operations: with stochastic mask.
  - crossover probability: $P_c = 0.85$;
  - mutation probability: $P_m = 0.08$ or $P_m = P_c$ if crossover is not applied;
  - inversion probability: $P_i = 0.02$;
  - reproducers selection method: Boltzmann's law inspired [2.7], with adaptive annealing temperature [2.34];
  - survivors selection: elitist generational with proportion $P_e = 10\%$ of the population;

- control parameter of the exploration-exploitation trade-off: $\delta = 0$ (the user controls this compromise by means of the annealing temperature);
- stop threshold with respect to residual energy: $\varepsilon = 0.001$;
- maximum number of generations during the evolution: $N_g = 100$;
- survival factor: $S = 5$ for the first 5 generations and $S = 3$ for the remaining generations.

➢ Basic wavelet to build the time-frequency-scale dictionary: g (for example the one defined in [2.25]).

➢ Scale parameter in the dictionary: $\sigma_0 = 1/2$.

2. Initialization.

a. Configure the time-frequency-scale dictionary, starting from the vibration features (selecting or estimating the parameters: t_0, σ, τ_0, ω_0, $M_{\max}$, $\left\{K_{m,\max}\right\}_{m\in\overline{0,M_{\max}}}$).

b. Construct the initial population for each dictionary scale, by using the uniform padding technique: $\left\{\boldsymbol{\mathcal{P}}_{m,0}^{u}\right\}_{m\in\overline{0,M_{\max}}}$.

c. Select the initial residual: $\Delta^0 x \equiv v$.

d. Initialize the residual index: $q = 0$.

3. For $q \geq 0$, construct the best matching atom:

3.1. If $\left\|\Delta^q x\right\| < \varepsilon$ stop the searching, since the vibration model has been determined. Jump to the final step, no. 4.

3.2. For each scale index $m \in \overline{0, M_{\max}}$:

3.2.1. If $q > 0$, choose the initial population $\boldsymbol{\mathcal{P}}_{m,0}^{q}$, in order to start the evolution. (If m is small enough, one can start from the population $\boldsymbol{\mathcal{P}}_{m,0}^{u}$; if m is sufficiently large, the initial population is the final population of the previous residual, $\boldsymbol{\mathcal{P}}_{m}^{q-1}$.)

3.2.2. Initialize the generation index: $p = 0$.

3.2.3. Estimate the best chromosome in the initial population $\boldsymbol{\mathcal{P}}_{m,0}^{q}$, denoted by $\gamma_{m,\max}^{q}$, and initialize its survival factor, $s = 0$.

3.2.4. While the evolution can continue ($p \leq N_g$) or the best chromosome has not yet reached its maximum survival factor ($s \leq S$), do:

3.2.4.1. Configure the elite of the population $\boldsymbol{\mathcal{P}}_{m,p}^{q}$ (select the best $N_e = \lfloor P_e N \rfloor$ chromosomes) and transfer the elite to the next population, $\boldsymbol{\mathcal{P}}_{m,p+1}^{q}$.

3.2.4.2. Build the transient population $\mathcal{R}_{m,p}^{q}$ (with $N - N_e$ chromosomes) by means of BGA, by using the Boltzmann's law and adaptive annealing temperature. (Some chromosomes in $\mathcal{R}_{m,p}^{q}$ could be represented by a certain number of clones.)

3.2.4.3. Initialize the number of applied genetic operations: $ng = 0$.

3.2.4.4. While $ng < N_{go}$ and the next population, $\mathcal{P}_{m,p+1}^{q}$, is incomplete, do:

a. Construct the masks of the genetic operations by using a P-PRSG based on BGA, with the probability profile [2.33]. The masks are built for each one of the two genes. This step is repeated until the crossover masks are both nulls or non nulls.

b. If all the masks are nulls, rebuild the mutation and the inversion masks with a P-PRSG and BGA, by using a modified probability profile (that excludes the crossover) :

$$\mathcal{P} = \begin{bmatrix} 1-P_c & P_c & 1-P_i & P_i \\ b_{m,0} & b_{m,1} & b_{i,0} & b_{i,1} \end{bmatrix}.$$

(Recall that, if the crossover is avoided, the mutation probability increases to value of the crossover probability.) This operation is repeated until a non null mask is obtained.

c. If the two crossover masks are non nulls, use a U-PRSG for selecting 2 different parents of the population $\mathcal{R}_{m,p}^{q}$ and apply the crossover, in order to obtain the 4 children. Make the children viable. Retain the 2 best different chromosomes in the set of 6 involved chromosomes and increment the number of the genetic operations: $ng \leftarrow ng + 1$.

d. If at least a mutation mask is non null, use a U-PRSG to choose a chromosome in the population $\mathcal{R}_{m,p}^{q}$ and apply the mutation. Make the mutant viable. Retain the best chromosome of the 2 involved chromosomes and increment the number of genetic operations: $ng \leftarrow ng + 1$.

e. If at least an inversion mask is non null, use a U-PRSG to select a chromosome in the population $\mathcal{R}_{m,p}^{q}$ and apply inversion. Make the mutant viable. Retain the best chromosome of the 2 involved chromosomes and increment the number of genetic operations: $ng \leftarrow ng+1$.

f. From all retained chromosomes, select the ones that do not yet belong to the next population $\mathcal{P}_{m,p+1}^{q}$ and add them to this population.

3.2.4.5. If the next population $\mathcal{P}_{m,p+1}^{q}$ is not complete, use a U-PRSG to generate the rest of the chromosomes, after making them viable.

3.2.4.6. Increment the generation index: $p \leftarrow p+1$.

3.2.4.7. Determine the best chromosome of the (new) population $\mathcal{P}_{m,p}^{q}$. If it has not changed ($\gamma_{m,\max}^{q}$), increment its survival factor: $s \leftarrow s+1$. Otherwise, a new optimal chromosome $\gamma_{m,\max}^{q}$ is detected and its survival factor has to be reset: $s=0$.

3.2.5. Denote $\mathcal{P}_{m}^{q}$ the final population and γ_{m}^{q} its best chromosome.

3.3. Determine the best chromosome in the set $\left\{\gamma_{m}^{q}\right\}_{m\in\overline{0,M_{\max}}}$ and denote m_q, n_q, k_q the optimal corresponding indices.

3.4. Retain the best wavelet coefficient $w^q \equiv \left\langle \Delta^q x, g_{\left[m_q,n_q,k_q\right]} \right\rangle$ and the best matching atom: $g^q \equiv w^q g_{\left[m_q,n_q,k_q\right]}$.

3.5. Estimate the next residual: $\Delta^{q+1}x \equiv \Delta^q x - g^q$.

3.6. Proceed with the next iteration: $q \leftarrow q+1$.

4. Return:

- The number of the best matching atoms: $Q=q$.
- The optimal wavelet coefficients: $\left\{w^i\right\}_{i\in\overline{0,Q-1}}$.
- The indices that pointing to the best matching atom positions in the dictionary $\left\{\left[m_i,n_i,k_i\right]\right\}_{i\in\overline{0,Q-1}}$.
- The vibration model: $v_D \equiv \sum_{q=0}^{Q-1} g^q$.

The resulted wavelet coefficients are now used in fault detection and diagnosis. Their magnitudes are shown for each scale in the time-frequency plane. A *time-frequency-scale map* is thus obtained. When comparing the map of a "healthy" (defect free) bearing with the map of a bearing (from the same family) that was affected by at least one defect, one notices that the wavelet coefficients are grouped in areas that can easily be associated to the fault types (even in case of multiple defects). Besides, the magnitude of the wavelet coefficients is a measure of defect severity, allowing one to estimate the *mean time before (total) failure* (MTBF). This is an important parameter helping the user to decide on the right instant the bearing should be replaced. The simulation results for this application are detailed in [STE 03a] and [STE 03b].

In general, the design of a "good" AG is difficult to manage. Although the implementation details of Algorithm 2.1 were not provided, this procedure is quite complex. A more elaborated version could be designed, in order to better control the exploration-exploitation trade-off.

In some applications, the GA are employed to define the zone (or vicinity) where the global optimum is most likely to be found. The remaining populations are used on this purpose. Once the area being delimited, another local optimization algorithm is used to refine the search. But in such cases, the fitness must be checked against certain properties of regularity. If this criterion is very irregular, then it is better to employ a global metaheuristic.

Despite the high complexity of the GA in this example, there are many applications where simplified genetic procedures have successfully been implemented. For example, the genetic algorithms are particularly well suited for optimization problems in planning and scheduling, as shown in [MES 98], [ZRI 08] and [MES 99].

2.3. Hill climbing by evolutionary strategies

There are situations in which the GA is too complex to be implemented. For example, if the fitness computation takes too long, it is suitable to reduce the number of criterion evaluations.

In this section, three algorithms close to GA family, but of reduced complexity, are presented. Since the fitness has to be maximized, the algorithms naturally belong to the category of hill climbing techniques. Unlike the techniques described in Section 1.3, here the peak is approached through a population capable of producing only mutant climbers.

The framework is the following. The criterion to maximize is [1.1], as usual. Actually, the criterion is seen here as the altitude indicator (altimeter) of an irregular mountain, with several peaks. To reach for one of the peaks (preferably the highest one), a group of N climbers is contracted, $\mathcal{A} \subset \mathcal{S}$. The position of each climber is defined as a chromosome, by a sequence of M symbols grouped into genes.

Frequently, in this context, the climbers are binary representations of the numbers in the search space $\mathcal{S}$. They verify a special property: only mutations can make the population evolve across the generations. Moreover, a mutant climber is produced by changing one symbol. If the climbers are not binary, the mutation of a symbol has to be defined. To facilitate understanding of the following algorithms, the chromosomes are symbol strings or binary representations.

Since the progress of the climbers to the peak can easily become slow for this type of mutations, a maximum number $P_{\max}$ of altimeter evaluations is enforced.

2.3.1. *Climbing by the steepest ascent*

The strategy of each climber is to determine its mutants to use the steepest slope. If the path has a dead end, the last mutant climber is abandoned and another climber "jumps" on a different starting point, even in case the point is located below the abandoned climber. The new climber also employs the steepest slope, hoping to arrive at a higher altitude than the one reached by his predecessor. The climbing stops either when the maximum number of altimeter evaluations is reached or all climbers are blocked. The optimum point is indicated by the climber in the group who is located at the highest altitude. The Algorithm 2.2 is based on this strategy.

Algorithm 2.2. *Hill climbing procedure by approaching the steepest ascent.*

1. Input data :
- Search space $\mathcal{S}$ (equations allowing the user to decide whether a point belongs or not to this set).
- Optimization criterion to maximize, f, seen as altimeter.
- Number of the climbers in the population, $N \in \mathbb{N}^*$.
- Number of necessary symbols to represent the position of each climber, $M \in \mathbb{N}^*$.
- Maximum number of altimeter evaluations, $P_{\max} \gg N$.

2. Initialization.

a. Select the starting positions of N climbers, $\mathcal{A} \subset \mathcal{S}$ (at random or by a certain strategy).

b. Initialize the group of the blocked climbers: $\mathcal{B} = \varnothing$ (void).

c. Initialize the altimeter evaluation index: $p = 0$.

3. While the group $\mathcal{A}$ includes at least one climber ($\mathcal{A} \neq \varnothing$) and $p \leq P_{\max} - \#\mathcal{A}$ (where $\#\mathcal{A}$ is the number of the climbers), do:

3.1. Use a U-PRSG to select a climber $a^0 \in \mathcal{A}$ (which constitutes a possible parent of several mutants).

3.2. Estimate its altitude, $f\left(a^0\right)$.

3.3. Increase the evaluation index : $p \leftarrow p+1$.

3.4. If $p > P_{\max} - \#\mathcal{A}$, move the climber a^0 into the group $\mathcal{B}$ and go to step 4.

3.5. Select a^0 as the temporary maximal point ($a_{\max}$) and $f\left(a^0\right)$ as the temporary maximal altitude ($f_{\max}$).

3.6. For $m \in \overline{1,M}$:

3.6.1. Apply mutation on the m-th symbol of climber a^{m-1}, in order to produce the mutant a^m, which will probably advance towards the peak.

3.6.2. Estimate the mutant altitude, $f\left(a^m\right)$, if $a^m \in \mathcal{S}$. Otherwise, set $f\left(a^m\right)=0$.

3.6.3. If $f\left(a^m\right) > f_{\max}$, the mutant indeed advanced and then:

3.6.3.1. Replace the temporary maximum: $a_{\max} \leftarrow a^m$ and $f_{\max} \leftarrow f\left(a^m\right)$.

3.6.3.2. Update $a^0 \leftarrow a^m$ in $\mathcal{A}$ group.

3.6.4. If $a^m \in \mathcal{S}$, increment the evaluation index: $p \leftarrow p+1$.

3.6.5. If $p > P_{\max} - \#\mathcal{A}$, move the climber a^0 into the group $\mathcal{B}$ and go to step 4.

3.7. If a^0 has not changed, there is a blocked climber on a dead end path. Move a^0 into the group $\mathcal{B}$.

4. Estimate the altitude of the climbers in group $\mathcal{A}$, if any. (The altitudes of blocked climbers in group $\mathcal{B}$ are already evaluated.)

5. Determine the climber in $\mathcal{A}$ and $\mathcal{B}$ located at the maximum altitude: $a^{\max}$. Its altitude is then $f^{\max} = f\left(a^{\max}\right)$.

6. Return:

- The current maximal point: $a^{\max}$.
- The current maximal altitude: $f^{\max}$.

Changing one symbol at a time produces, in general, a fairly rapid ascent on a steep ascent route, even if it leads to a local peak. Since this is a group of climbers, those who are blocked on local peak expect the other colleagues to better perform. However, the mutants go relatively quickly away from their parent (a^0), which

could make them miss the maximum points. For example, if $a^0 = 0 = 00000000$ (with $N = 8$), then the successive mutants are: $1 = 00000001$, $3 = 00000011$, $7 = 00000111$, $15 = 00001111$, $31 = 00011111$, $63 = 00111111$, $127 = 01111111$, $255 = 11111111$. The distance between the mutants is becoming larger and larger and the population can become full of lacunas. The maximum point can therefore hide in such a lacuna.

This algorithm is useful especially for a quick exploration of a large enough search space, the exploitation being less important.

2.3.2. *Climbing by the next ascent*

In order to reduce the gaps between mutants (as previously shown), an alternative strategy is used. The ascent could be performed gently, by using slopes that are quite close to each other. The following procedure in Algorithm 2.3 is similar to the previous one, but the mutants are generated in a different way.

Algorithm 2.3. *Hill climbing procedure by approaching the next ascent.*

1. Input data :
- Search space $\mathcal{S}$ (equations allowing the user to decide whether a point belongs or not to this set).
- Optimization criterion to maximize, f, seen as altimeter.
- Number of the climbers in the population, $N \in \mathbb{N}^*$.
- Number of necessary symbols to represent the position of each climber, $M \in \mathbb{N}^*$.
- Maximum number of altimeter evaluations, $P_{\max} \gg N$.

2. Initialization.
a. Select the starting positions of N climbers, $\mathcal{A} \subset \mathcal{S}$ (at random or by a certain strategy).
b. Initialize the group of the blocked climbers: $\mathcal{B} = \varnothing$ (void).
c. Initialize the altimeter evaluation index: $p = 0$.

3. While the group $\mathcal{A}$ includes at least one climber ($\mathcal{A} \neq \varnothing$) and $p \leq P_{\max} - \#\mathcal{A}$ (where $\#\mathcal{A}$ is the number of the climbers), do:

3.1. Use a U-PRSG to select a climber $a^0 \in \mathcal{A}$ (which constitutes a possible parent of several mutants).

3.2. Estimate its altitude, $f\left(a^0\right)$.

3.3. Increase the evaluation index: $p \leftarrow p + 1$.

3.4. If $p > P_{\max} - \#\mathcal{A}$, move the climber a^0 into the group $\mathcal{B}$ and go to step 4.

3.5. Select a^0 as the temporary maximal point ($a_{\max}$) and $f\left(a^0\right)$ as the temporary maximal altitude ($f_{\max}$).

3.6. For $m \in \overline{1,M}$:

3.6.1. Apply mutation on the m-th symbol of climber a^0, in order to produce the mutant a^m, which will probably advance towards the peak.

3.6.2. If $a^m \in \mathcal{S}$, estimate the mutant altitude, $f\left(a^m\right)$. Otherwise, set $f\left(a^m\right)=0$.

3.6.3. If $f\left(a^m\right) > f_{\max}$, the mutant indeed advanced and then:

3.6.3.1. Replace the temporary maximum: $a_{\max} \leftarrow a^m$ and $f_{\max} \leftarrow f\left(a^m\right)$.

3.6.3.2. Update $a^0 \leftarrow a^m$ in $\mathcal{A}$ group.

3.6.4. If $a^m \in \mathcal{S}$, increase the evaluation index: $p \leftarrow p+1$.

3.6.5. If $p > P_{\max} - \#\mathcal{A}$, move the climber a^0 into the group $\mathcal{B}$ and go to step 4.

3.7. If a^0 has not changed, there is a blocked climber on a dead end path. Move a^0 into the group $\mathcal{B}$.

4. Estimate the altitude of the climbers in group $\mathcal{A}$, if any. (The altitudes of blocked climbers in group $\mathcal{B}$ are already evaluated.)

5. Determine the climber in $\mathcal{A}$ and $\mathcal{B}$ located at the maximum altitude: $a^{\max}$. Its altitude is then $f^{\max} = f\left(a^{\max}\right)$.

6. Return:

- The current maximal point: $a^{\max}$.
- The current maximal altitude: $f^{\max}$.

Returning to the previous example, since in this algorithm the step 3.6.1 is different, if $a^0 = 0 = 00000000$ (with $N=8$), then the successive mutants are: $1 = 00000001$, $2 = 00000010$, $4 = 00000100$, $8 = 00001000$, $16 = 00010000$, $32 = 00100000$, $64 = 01000000$, $128 = 10000000$. The gaps are thus reduced and the ascent can be smoother than in the previous algorithm.

This algorithm emphasizes the exploitation, although exploration still is more important.

2.3.3. *Hill climbing by group of alpinists*

In Algorithm 2.4 below, the climbers approach the mountain in group. After winning a new position, each climber is waiting for the others to occupy their new positions as well.

Algorithm 2.4. *Hill climbing by group of alpinists.*

1. Input data :
- Search space $\mathcal{S}$ (equations allowing the user to decide whether a point belongs or not to this set).
- Optimization criterion to maximize, f, seen as altimeter.
- Number of the climbers in the population, $N \in \mathbb{N}^*$.
- Number of necessary symbols to represent the position of each climber, $M \in \mathbb{N}^*$.
- Maximum number of altimeter evaluations, $P_{\max} \gg N$.

2. Initialization.

a. Select the starting positions of N climbers, $\mathcal{A} \subset \mathcal{S}$ (at random or by a certain strategy). Thus: $\mathcal{A} = \{a_n\}_{n \in \overline{1,N}}$.

b. Use a U-PRSG to select a climber a of group $\mathcal{A}$ and initialize the current solution to: $\left\{a^{\max} = a,\ f^{\max} = f(a)\right\}$.

c. Initialize the altimeter evaluation index: $p = 0$.

3. While $p \le P_{\max}$ **, do:**

3.1. Use a U-PRSG to select the position of symbol to be changed by the mutation: $m \in \overline{1,M}$.

3.2. For $n \in \overline{1,N}$:

3.2.1. Apply mutation on the m-th symbol of climber a_n, in order to produce the mutant a_n^m, who will probably advance towards the peak.

3.2.2. If $a_n^m \in \mathcal{S}$, estimate the mutant altitude, $f\left(a_n^m\right)$. Otherwise, set $f\left(a_n^m\right) = 0$.

3.2.3. If $f\left(a_n^m\right) > f^{\max}$:

3.2.3.1. Update the information concerning the maximum: $a^{\max} \leftarrow a_n^m$, $f^{\max} \leftarrow f\left(a_n^m\right)$.

3.2.3.2. Replace the climber a_n by his mutant a_n^m in the group $\mathcal{A}$.

3.2.4. If $a_n^m \in \mathcal{S}$, increment the evaluations index: $p \leftarrow p+1$.

3.2.5. If $p > P_{\max} - 1$, go to the final step, no. 4.

4. Return:

- The current maximal point: $a^{\max}$.
- The current maximal altitude: $f^{\max}$.

This procedure can be improved such that the climbers never produce the same mutant twice on their way to the top. The algorithm reaches to an acceptable exploration-exploitation trade-off, but it can become lengthy.

2.4. Optimization by ant colonies

2.4.1. *Ant colonies*

2.4.1.1. *Natural ants*

Biologists have noticed that ants are quickly able to find the shortest path from the nest to a food source. The explanation is as follows:

- ants go in randomly chosen directions, each one laying a pheromone trail on its path;
- as soon as one of them found the food, it returns to the nest, depositing pheromone again;
- the ants prefer to follow the paths with the highest concentration of pheromone;
- consequently, the new ants who leave the nest will tend to follow the paths that other ants returned to the nest (with food).

This approach could rapidly lead to a local minimum, but the pheromone gradually evaporates and some ants systematically are thus enforced to explore new paths. Therefore, this problem is avoided. After a while, the overwhelming majority of ants are following the shortest path, as shown in Figure 2.15.

In the animal world, this type of information exchange by modifying the environment, referred to as *stigmergy*, is very often encountered. The French biologist Pierre-Paul Grassé introduced this concept in 1959, in order to reflect the termite behavior. Thus, according to Grassé, the stigmergy is a phenomenon of "workers stimulation by the performance they have achieved". (The term actually comes from Greek and means "mark the work".) Thus, the environment is marked by the some insects with fluids (of chemical nature) on their paths. The insects not only they mark the way towards the food, but also they define a certain territory or try to attract the reproducers.

The behavior of the ants looking for food shows very well the mechanism of natural optimality related to finding the shortest path in a graph. Inspired by this behavior, various algorithms were developed.

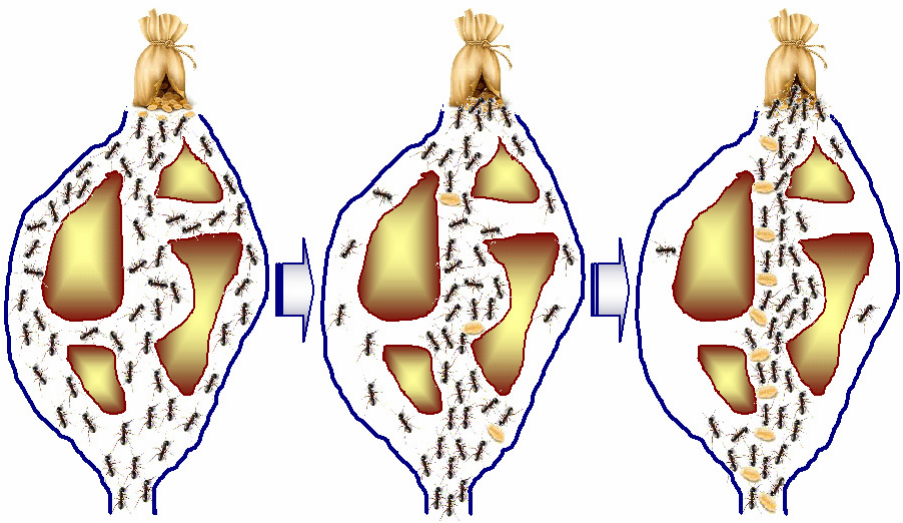


Figure 2.15. *Illustration of natural ant colony behavior leading to the optimality.*

2.4.1.2. *Aspects inspired from natural ants*

The ant colony allows defining some concepts that could contribute to the design of a metaheuristic algorithm, known under the generic name of *Ant Colony Algorithm* (ACA). The first ACA was proposed by Marco Dorigo in his PhD thesis from 1992 [DOR 92] (the basic idea being published in [COL 92], shortly before the public defense).

The observations below express the principles founding the ACA:

- To configure an ACA, a cooperative population of agents referred to as *ant colony* is implemented.
- The agents lay pheromone traces to communicate their trails (by stigmergy). The trails constitute a memory of long-term experience accumulated by all agents.
- The evaporation is taken into account. This phenomenon allows avoiding a premature convergence to a local optimum and contributes to search diversifying.
- The solution gradually is build from a set of local paths.
- To move towards an optimal solution, the strategy of stochastic movement is adopted. Thus, a path is selected at random, according to a predefined probability, starting from the set of the pheromone deposits.
- The population update is guided by the quality of the solutions. In fact, for the most ants, the amount of spread pheromone on the way back to the nest is as important as the food source.

2.4.1.3. *Features developed for the artificial ants*

ACA belong to a class of metaheuristics that simulate the *swarm intelligence* [BON 99], [PAR 11]. (The stigmergy actually is a key concept of swarm intelligence.) The artificial ants have the following characteristics:

- each ant has its own memory to keep in mind the traveled paths and to evaluate the current solution for which the pheromone trace has to be strengthened in the end of its journey;
- the pheromone trace can be updated each time an ant has traveled a full itinerary, by accounting the quality of the found solutions;
- the ants could have the capacity to facilitate the exploration on paths that have not been considered yet;
- the ants move in a graph, along arcs with labels set by the intensity of pheromone traces left by other traveling ants.

2.4.2. *Basic optimization algorithm by ant colonies*

The optimization problem [1.2], which can be solved in context of this section, is enforced to exhibit two main features, as suitable for ant colony:

a. The search space $\mathcal{S}$ is discrete (i.e. granular) and organized as oriented graph, with the following characteristics:
   - a node can have one or more inheritors (*children*); in this case, the node stands for the *parent*; at maximum, for any point of $\mathcal{S}$, all the remaining points of $\mathcal{S}$ are its children (which means the graph becomes a *complete network*);
   - the graph could have *terminal* nodes (without children), also referred to as *leaves*;
   - the oriented arcs determine the direction of movement from one node to another; additionally, each arc has a memory to store information about the trails made by the agents of a population (like artificial ants);
   - the distance between the parent and its child could a priori be known, in which case, the corresponding arc is labeled accordingly;
   - each passage from a node to another could involve a cost, also marked on the corresponding arc (if necessary).

b. The optimization criterion f expresses either the traveled distance on a path in the graph (between two nodes), the cost of this route, or a combination of both.

The problem is to minimize the criterion f, which means to find either the shortest path or the least expensive one of the search space $\mathcal{S}$. Finding a path which is at the same time short and inexpensive could constitute an optimization problem in this framework as well.

If the search space cannot be organized as oriented graph with the above mentioned properties, the ACA is not appropriate as optimization tool. But, in general, there are few cases where the distances and/or price to be paid between the points of $\mathcal{S}$ cannot be defined. It is true that the greater the number of children of a parent, the less effective the ACA (since the computational burden of the optimization problem can rapidly increase).

In case of the ant colony, Figure 2.16 below shows the variables and the characteristic parameters of the arc parent-child, which could be traveled by an ant.

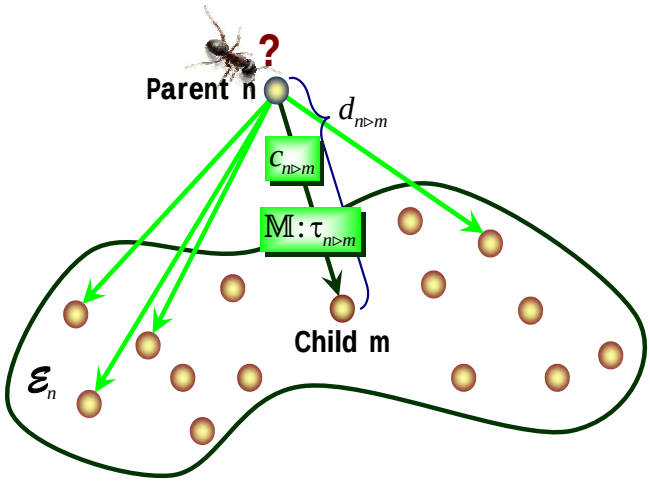


Figure 2.16. *Parameters and variables in a graph associated to ACA.*

Denote by $\mathcal{E}_n$ the set of children corresponding to parent $n \in \mathcal{S}$. For any child $m \in \mathcal{E}_n$, let $n \triangleright m$ stand for the transition parent-child. The same notation is used to mark the arc between m and n. The distance between these two nodes actually represents the length of the arc, $d_{n \triangleright m}$. On the arc, one can introduce a cost to pay, $c_{n \triangleright m}$, when traveling along the route $n \triangleright m$. At least one of the two parameters $d_{n \triangleright m}$ or $c_{n \triangleright m}$ must figure as arc label, otherwise the ACA can not be designed. The memory $\mathbb{M}$ of the arc contains the intensity of current pheromone trace, $\tau_{n \triangleright m}$. This parameter is determined not only by all the ants in the colony that have traveled from m to n and left some pheromone, but also by evaporation.

Denote by $\mathcal{F} = \{f_p\}_{p \in \overline{1,P}}$ a colony of $P \in \mathbb{N}^*$ (artificial) ants that explore the search space $\mathcal{S}$. When some ant f in the colony arrived in the node $n \in \mathcal{S}$, it faces the dilemma of choosing the next path to follow towards one of the children bond to that node. How to choose the path then? The ant solves this problem in a stochastic manner, by using a probability density associated to the set of reachable

children $\mathcal{E}_n$, that will be built in ad hoc manner. Thus, the probability that the ant chooses the child $m \in \mathcal{E}_n$ as its next target is computed as follows (in the most general case, where the arc is characterized by both distance and cost):

$$\wp_{n \triangleright m} = \frac{\tau_{n \triangleright m}^{\alpha} \eta_{n \triangleright m}^{\beta} \varphi_{n \triangleright m}^{\gamma}}{\sum_{l \in \mathcal{E}_n} \tau_{n \triangleright l}^{\alpha} \eta_{n \triangleright l}^{\beta} \varphi_{n \triangleright l}^{\gamma}}, \qquad [2.35]$$

where:

- $\eta_{n \triangleright m}$ is an indicator depending on the distance $d_{n \triangleright m}$;
- $\varphi_{n \triangleright m}$ is another indicator, expressed in terms of cost $c_{n \triangleright m}$;
- $\alpha \geq 0$, $\beta \geq 1$, $\gamma \geq 1$ are weights expressing the importance of the pheromone trail $n \triangleright m$, of the $\eta_{n \triangleright m}$ indicator and of the $\varphi_{n \triangleright m}$ indicator, respectively.

Usually, the definitions of the two indicators in expression [2.35] are:

$$\eta_{n \triangleright m} = \frac{1}{d_{n \triangleright m}}; \quad \varphi_{n \triangleright m} = \frac{1}{c_{n \triangleright m}}. \qquad [2.36]$$

In other words, the probability to travel towards a child is inversely proportional to the distance with respect to the parent and the traveling cost. If the cost is not specified, by convention $\varphi_{n \triangleright m} = 1$. Similarly, if the distance is not specified, then $\eta_{n \triangleright m} = 1$.

The important parameters are freely chosen by the user, in order to obtain a good exploration-exploitation trade-off for the ant colony.

By using the BGA and the probability density [2.35] (eventually with a higher resolution), the ant f chooses a target child, for example $m \in \mathcal{E}_n$. On its path, it lays a pheromone trace. Consequently, the pheromone intensity $\tau_{n \triangleright m}$ changes. A formula for the trace intensity update is as follows:

$$\tau_{n \triangleright m} \leftarrow (1 - \rho)\tau_{n \triangleright m} + \sum_{f \in \mathcal{F}_{n \triangleright m}} \Delta\tau_{n \triangleright m}(f), \qquad [2.37]$$

where:

- $\rho \in (0,1)$ is the pheromone evaporation factor;
- $\Delta\tau_{n \triangleright m}(f)$ is the amount of pheromone deposited by ant f on the trail $n \triangleright m$;
- $\mathcal{F}_{n \triangleright m}$ is the set of the ants from colony $\mathcal{F}$ who traveled on the arc $n \triangleright m$ at the same searching stage.

Certainly, if no ant is going on the arc $n \triangleright m$ (i.e. if $\mathcal{F}_{n \triangleright m} = \varnothing$), then the formula [2.37] shows that the pheromone trace on this route gradually evaporates (since $0 < 1 - \rho < 1$).

The amount of pheromone the ant f could lay on its trail depends on the natural capacity of the colony, quantified by a constant $Q > 0$, which is freely chosen by the user. This amount gradually decreases, as the ant travels across more and more nodes in the graph. More specifically, one can write:

$$\Delta\tau_{n \triangleright m}(f) = \frac{Q}{f_{0 \triangleright \cdots \triangleright n \triangleright m}(f)}, \qquad [2.38]$$

where $f_{0 \triangleright \cdots \triangleright n \triangleright m}(f)$ is the value of the f criterion evaluated for the traveling path $0 \triangleright \cdots \triangleright n \triangleright m$ of ant f, from its departure up to the node m. Before starting the journey, the pheromone tank of any ant is full. Normally, since the criterion f is proportional to the sum of the traveled distances (or to the total paid price, or to both of them), definition [2.38] shows that, during its journey, the ant has less and less pheromone in the tank to mark its trails.

Naturally, the colony starts the exploration with randomly distributed ants on the search space (in fact, on the graph nodes). In order to start the search, a small pheromone trace has to be laid on all arcs of the graph. The search stops either when all the ants arrived at the graph leaves (if any), or after a maximum number of iterations (or of parent-children traveled paths), $K_{\max}$, has been reached.

The optimal solution of problem [1.2] is indicated by the ant with the highest performance $f_{0 \triangleright \cdots \triangleright n}(f)$. Note that the ant is endowed with a memory to store its current path.

The basic procedure of optimization by means of ant colonies is summarized in Algorithm 2.5.

Algorithm 2.5. *Basic procedure for optimization though ant colony.*

1. **Input data :**
   - Search space $\mathcal{S}$, organized as an oriented graph, having all the distances and/or the transition costs well defined for all the arcs.
   - Ant performance f (definition of the criterion to minimize). Usually, f is computed for the paths in the search space.
   - Configuring parameters:
     - the number of ants in the colony, $P \in \mathbb{N}^*$;
     - the initial amount of pheromone carried by each ant in the colony, $Q > 0$;

- the initial amount of pheromone to label all arcs in the search space, $\tau_0 > 0$ (usually, τ_0 is a small fraction of Q);
- the definition of distance indicator, $\eta_{n \triangleright m}$; by default, if the distances are specified, $\eta_{n \triangleright m} = 1/d_{n \triangleright m}$; otherwise, $\eta_{n \triangleright m} = 1$;
- the definition of cost indicator, $\varphi_{n \triangleright m}$; by default, if the costs are specified, $\varphi_{n \triangleright m} = 1/c_{n \triangleright m}$; otherwise, $\varphi_{n \triangleright m} = 1$;
- the importance weights: $\alpha \geq 0$, $\beta \geq 1$ and $\gamma \geq 1$; by default: $\alpha = \beta = \gamma = 1$;
- the pheromone evaporation factor, $\rho \in (0,1)$.

➢ Stop test. For example, the maximum number of iterations to perform, $K_{\max} \in \mathbb{N}^*$ (even if there are some leaves in the graph).

2. Initialization.

a. Denote by $\mathcal{F} = \{f_p\}_{p \in \overline{1,P}}$ the ant colony. Choose at random or by a certain strategy the ants departure positions, $\mathcal{N} = \{n_0^f\}_{f \in \mathcal{F}} \subset \mathcal{S}$. Some positions could repeat, which means that several ants could start their journeys from the same node.

b. Initialize the memory of each ant with the departure point of its path (i.e. with the corresponding positions of $\mathcal{N}$).

c. Initialize the set of ants that arrived on leaves of the graph (being thus blocked): $\mathcal{B} = \varnothing$.

d. Choose the amount of pheromone to label all the arcs of the search space, $\tau_0 > 0$, as a small fraction of Q. Set this amount in the memory of all arcs.

e. Initialize the number of iterations: $k = 0$.

3. While $\mathcal{F} \neq \varnothing$ and $k \leq K_{\max}$, do:

3.1. For each ant $f \in \mathcal{F}$:

3.1.1. Determine the set $\mathcal{E}_n(f)$ of children bond to the node $n \in \mathcal{S}$, which currently is hosting the ant. This set only includes the nodes that the f ant could reach.

3.1.2. If $\mathcal{E}_n(f) = \varnothing$, then the ant is blocked on a leaf. In this case, it is removed from the colony ($\mathcal{F} \leftarrow \mathcal{F} \setminus \{f\}$), being added to the set of blocked ants ($\mathcal{B} \leftarrow \mathcal{B} \cup \{f\}$).

3.1.3. Otherwise:

3.1.3.1. Build the probability density by using definition [2.35]. (Initially, $\tau_{n \triangleright m} = \tau_0$, $\forall m \in \mathcal{E}_n(f)$.)

3.1.3.2. With this probability density, use a P-PRSG (based on BGA) to choose the target child $m \in \mathcal{E}_n(f)$. (Choose a sufficient resolution, in order to avoid missing children.)

3.1.3.3. Move the ant towards the child m previously selected.

3.1.3.4. Evaluate the ant performance, $f_{0 \triangleright \cdots \triangleright n \triangleright m}(f)$.

3.1.3.5. If the ant is the first one that traveled the arc $n \triangleright m$ at the current iteration k, initialize $\mathcal{F}_{n \triangleright m} = \{f\}$. Otherwise, the ant will join the set of the other ants that traveled along the trail $n \triangleright m$ at the current iteration: $\mathcal{F}_{n \triangleright m} \leftarrow \mathcal{F}_{n \triangleright m} \cup \{f\}$.

3.1.3.6. Evaluate the amount of laid pheromone on the trail, $\Delta\tau_{n \triangleright m}(f)$, by using definition [2.38].

3.2. If $\mathcal{F} \neq \varnothing$:

3.2.1. For each trail $n \triangleright m$ of step 3.1, update the pheromone trace by means of equation [2.37].

3.2.2. For all parent-child trial of type $n \triangleright m$, simulate the evaporation phenomenon, as follows:

$$\tau_{n \triangleright m} \leftarrow \max\{(1-\rho)\tau_{n \triangleright m}, \tau_0\}.$$

3.2.3. Proceed with the next iteration: $k \leftarrow k+1$.

4. Choose the best ant in the set $\mathcal{F} \cup \mathcal{B}$.

5. Return:

- The optimal path, corresponding to the best ant.
- The optimal criterion value on the best ant path.

The configuration parameters have a crucial impact on the performance of Algorithm 2.5 (especially on the exploration-exploitation trade-off). Starting from the involvement of ACA in practical applications, some interesting remarks can be formulated:

- if the ant colony is too large, the exploration in the search space is performed in small steps; therefore, ACA could become slow; the recommended values of the colony size P are between 10 and 50;
- if the weight α is too small (close to 0), the ants tend to exhaustively explore the search space, since the pheromone traces are not perceived with their real intensity; therefore, the ACA could become greedy;
- if the weight β or γ is too small (close to 1), the ants can quickly confine themselves with the first optimal path found; therefore, ACA could prematurely converge to a local minimum;

- if the evaporation factor ρ is too large (over 1/2), the ants quickly forget the optimal routes and tend to return on the same paths in a cyclic manner; consequently, the ACA could start oscillating; the best values of evaporation factor are between 0.01 and 0.2.

To avoid premature convergence of the algorithm, the intensity of pheromone traces can be bounded:

$$\tau_{n \triangleright m}(f) \in \left[\tau_{\min}, \tau_{\max}\right], \quad \forall f \in \mathcal{F}. \qquad [2.39]$$

Usually, $\tau_{\min} = \tau_0$ (the traces intensity in the beginning of exploration).

For certain ACA, the pheromone traces are initialized to the maximum value $\tau_{\max}$ (and not to the minimum value $\tau_{\min} = \tau_0$, like in Algorithm 2.5), in order to yield greater exploration in the beginning of search.

If the evaporation is too fast, the search can be strengthened by making the following adjustment, after updating the pheromone trace:

$$\tau_{n \triangleright m}(f) \leftarrow \lambda \tau_{n \triangleright m}(f) + (1-\lambda)\tau_{\max}, \quad \forall f \in \mathcal{F}, \qquad [2.40]$$

where $\lambda \in (0,1)$ usually is chosen to be small.

This type of ACA is based on the concept of *Max-Min Ant System* [STU 97], [STU 00]. It allows avoiding the bottlenecks on obvious suboptimal roads, while adjusting the balance between the ability of exploration and capacity of exploitation featured by the ant colony.

ACA proved to be fairly fast convergent in problems such as the traveling salesman, where each node has to be visited no more than once [DOR 97]. (This application is presented in the end of section.) If the graph includes cycles in its structure, then the search could take time. For this reason, a maximum number of iterations is imposed in Algorithm 2.5. However, if the configuration parameters are correctly selected, it is quite unlikely that all the ants return on the same path cyclically.

A delicate problem of ACA is the initialization, to which it is quite sensitive. If the graph contains a root, then the problem is simpler, since the initialization can be realized by starting either from the children bond to the root, or from other nodes provided that they are not too far away from the root. In this case, the ant performance along the path between the root and its departure node is added to the optimal solution performance. If the graph has no root, the departure points of the colony have to be chosen with care. Prior knowledge of the search space properties could be helpful in the attempt for initialization.

To the best of our knowledge, no satisfactory explanation for the convergence of ACA has been published yet. When comparing to GA, the ACA seemingly are more unstable in applications and can easily fail. However, their advantage over GA consists of smaller number of configuration parameters to set.

2.4.3. *Pheromone trail update*

After their introduction in the literature, the ACA evolved in several directions, also suggested by equations [2.35], [2.37] and [2.38]. Different definitions of the indicators $\eta_{n \triangleright m}$ and $\varphi_{n \triangleright m}$ are found in the literature [DOR 96], [DOR 99], [DOR 04], [MON 10]. Similarly, the α, β and γ weights could vary (perhaps adaptively during the search), in order to accomplish the best exploration-exploitation trade-off.

The pheromone trail is very important in the search for optima. Its update can be realized in various ways, with respect to equations [2.37] and [2.38]. Before revealing some alternate techniques for pheromone trail update, it should be emphasized that this track is continuously changing from iteration to iteration, but according to two possible situations during the exploration of the search space. Firstly, if the route from the parent node n to the child node m (i.e. $n \triangleright m$) is not used by an ant in the colony, the pheromone gradually evaporates. However, a small pheromone trace has to persist on the trail, in order not to completely remove it from the search space (see step 3.2.2 of Algorithm 2.5). Secondly, if the trail $n \triangleright m$ was employed by at least one ant in the colony, its trace has to be refreshed with a new amount of pheromone, although the evaporation phenomenon continues to occur (see the meaning of the equation [2.37]).

Equation [2.38] shows how the pheromone quantity laid by an ant can be determined, depending on the length of its route (the longer the road, the less laid pheromone). This leads to a delayed update of the trail.

Another evaluation manner of the laid pheromone quantity on the trail $n \triangleright m$, corresponding to the ant $f \in \mathcal{F}$, at current iteration $k \in \overline{1, K_{\max}}$, is explained next.

2.4.3.1. *Adaptive delayed update*

During the exploration, at current iteration, there is at least one ant in the colony having traveled a path of minimum length or cost: $f^{\min} \in \mathcal{F}$. Denote $f_{0 \triangleright \cdots \triangleright n \triangleright m}\left(f^{\min}\right)$ the performance of this ant. Then the numerator Q of equation [2.38] is replaced by $f_{0 \triangleright \cdots \triangleright n \triangleright m}\left(f^{\min}\right)$. It results:

$$\Delta\tau_{n \triangleright m}(f) = \delta \frac{f_{0 \triangleright \cdots \triangleright n \triangleright m}\left(f^{\min}\right)}{f_{0 \triangleright \cdots \triangleright n \triangleright m}(f)}, \qquad [2.41]$$

where $\delta \in (0,1)$ is a weight chosen by the user.

In this case, since both terms of the ratio in equation [2.41] increase during the search, the pheromone quantity is adaptively chosen. The ants that traveled too long or along too expensive routes compared to the best ant in the colony, will deposit a

small amount of pheromone on their route. On the contrary, equation [2.41] promotes the ants with similar performances to the best ant.

For this type of update, the exploration-exploitation balance is slanted towards exploitation whenever the exploration tends to dominate the search.

2.4.3.2. *On-line update*

Some ant species usually mark their paths by a constant amount of pheromone. In this case:

$$\Delta\tau_{n\triangleright m}(f) = \tau_{\mathcal{F}}\,, \qquad [2.42]$$

where $\tau_{\mathcal{F}} > 0$ is a specific constant of colony $\mathcal{F}$. Now, the exploration-exploitation trade-off rather is controlled through the weights α, β and γ.

2.4.3.3. *Update through elitist strategy*

Return to the ant $f^{\min} \in \mathcal{F}$, like in paragraph 2.4.3.1. This time, one considers that $f^{\min}$ is the best ant in the colony, taking into account that the route it traveled, denoted by $n_0^{\min} \triangleright \cdots \triangleright n_{k-1}^{\min} \triangleright n_k^{\min}$, led to the minimum of criterion [DOR 96]. This path is thus optimal. More specifically:

$$f_{n_0^{\min}\triangleright\cdots\triangleright n_{k-1}^{\min}\triangleright n_k^{\min}}\left(f^{\min}\right) \le f_{n_0^{f}\triangleright\cdots\triangleright n_{k-1}^{f}\triangleright n_k^{f}}(f)\,, \quad \forall f \in \mathcal{F}\,, \qquad [2.43]$$

with natural notations. Then, the pheromone trail of this ant is updated not only for the last part of the route $n_{k-1}^{\min} \triangleright n_k^{\min}$, but also for the whole route, in order to consolidate its position (which would otherwise be affected by evaporation).

Beside the update of pheromone quantity for the trail $n_{k-1}^{\min} \triangleright n_k^{\min}$, a supplementary amount of pheromone is added to the traces of some other routes with optimal path:

$$\Delta\tau_{n_{i-1}^{\min}\triangleright n_i^{\min}}\left(f^{\min}\right) = \frac{Q}{f_{n_0^{\min}\triangleright\cdots\triangleright n_{k-1}^{\min}\triangleright n_k^{\min}}\left(f^{\min}\right)}\,, \quad \forall i \in \overline{1,k}\,. \qquad [2.44]$$

Consequently, equation [2.37] leads to the following updating expression of the pheromone amount on optimal trail:

$$\tau_{n_{i-1}^{\min}\triangleright n_i^{\min}} \leftarrow (1-\rho)\tau_{n_{i-1}^{\min}\triangleright n_i^{\min}} + \rho\Delta\tau_{n_{i-1}^{\min}\triangleright n_i^{\min}}\left(f^{\min}\right), \quad \forall i \in \overline{1,k}\,. \qquad [2.45]$$

This update method is intended to build an elite among the explored paths and, thus, among the ants in the colony. Like in case of GA, the elitist strategy favors the exploitation to detriment of exploration.

2.4.3.4. *Update by ants ranking*

By means of this strategy, the ants are sorted in descending order, according to the performances of traveled paths (the best ants on first positions). This time, for each iteration, only the first P_e ants (over the P ants in the colony) lay pheromone on all the paths they have traveled.

The rank of an ant is equal to the position it takes in the $\mathcal{F}$ colony, after ordering. Thus, the best ant is of rank 1, the second-best ant has rank 2, etc. Denote by $r(f) \in \overline{1,P}$ the rank of ant $f \in \mathcal{F}$, while $n_0^{r(f)} \triangleright \cdots \triangleright n_{k-1}^{r(f)} \triangleright n_k^{r(f)}$ is the path it had traveled until the current iteration.

Thus, the pheromone trace update on the trails of the best P_e ants in the colony is performed as follows:

$$\tau_{n_{i-1}^r \triangleright n_i^r} \leftarrow (1-\rho)\tau_{n_{i-1}^r \triangleright n_i^r} + \frac{\rho}{P_e}\sum_{p=1}^{P_e}(P_e - p + 1)\Delta\tau^p_{n_{i-1}^r \triangleright n_i^r}, \quad \forall i \in \overline{1,k}, \ \forall r \in \overline{1,P_e}, \quad [2.46]$$

where:

$$\Delta\tau^p_{n_{i-1}^r \triangleright n_i^r} = \begin{cases} \dfrac{Q}{f_{n_0^p \triangleright \cdots \triangleright n_{k-1}^p \triangleright n_k^p}(f_p)} & \text{, if } f_p \text{ has traveled along } n_{i-1}^r \triangleright n_i^r \\ 0 & \text{, otherwise} \end{cases},$$

$$\forall i \in \overline{1,k}, \ \forall r, p \in \overline{1,P_e}. \quad [2.47]$$

The parameter P_e actually allows the user to control the elite size and, subsequently, the exploration-exploitation trade-off.

2.4.4. *Systemic ant colony algorithm*

An alternate procedure of Algorithm 2.5 was proposed in [DOR 97]. This version of ACA is based on a systemic view of the search performed by the ant colony. Instead of working with the nodes of a graph, one works now with the states of a dynamic system. In fact, there is no fundamental difference between the concepts of "node" and "state" from the point of view of search space internal organization. Nevertheless, in the new approach, the ants in the colony pass from one state to another, hoping to find the most efficient dynamic evolution to achieve the objective (minimizing a criterion). Certainly, as already stated before, an ant passing from one state to another has to pay for its journey. The ant is enforced

either to spend resources when moving between states, or to cover the traveling expenses (or both at the same time). Unlike the basic ACA, where the search rather is governed by the distance, here, the exploration engine seemingly is based on the traveling costs. The distance between the states is less intuitive than the transfer cost from one state to another.

The fundamental difference between the *systemic ant colony algorithms* (SACA), which will be described hereafter, and the basic ACA consists of how to select the child who will host the ant at the next iteration (i.e. the target child).

After touching a state $n \in \mathcal{S}$, the ant $f \in \mathcal{F}$ is selecting the target state following the strategy below:

a. Set a number $h \in [0,1]$ at random (but uniformly).
b. If h is smaller than some threshold h_s a priori known, minimize the criterion:

$$f_n[m] = \tau_{n\triangleright m}^{\alpha} \eta_{n\triangleright m}^{\beta} \varphi_{n\triangleright m}^{\gamma}, \quad \forall m \in \mathcal{E}_n, \qquad [2.48]$$

in order to find the target state. In [2.48], $\mathcal{E}_n$ is the set of all children the ant could reach for, when starting from the n state. (The notations of equation [2.35] are preserved in this framework too.)
c. Otherwise, use the probability density [2.35] and the BGA to select the target state.

The basic characteristics of ACA describe the SACA as well. The main steps of systemic procedure with ant colonies are presented within the Algorithm 2.6.

Algorithm 2.6. *Systemic optimization procedure by means of ant colony.*

1. Input data:

- Search space $\mathcal{S}$, organized as an oriented graph, having all the distances and/or the transition costs well defined for all the arcs between nodes, seen as states of some system.
- Ant performance f (definition of the criterion to minimize). Usually, f is computed for the paths in the search space.
- Configuring parameters:
  - the number of ants in the colony, $P \in \mathbb{N}^*$;
  - the elite size in the colony, $P_e \in \overline{1,P}$;
  - the maximum number of target states for each ant, $N_c \in \mathbb{N}^*$;
  - the initial amount of pheromone carried by each ant in the colony, $Q > 0$;
  - the pheromone trace intensity limits on trails from the search space, $0 < \tau_{\min} < \tau_{\max}$;

- the threshold of colony relative variance to restart the exploration of search space, $\varepsilon \in [0,1]$;
- the correction factor of pheromone trace intensity, $\lambda \in [0,1]$;
- the definition of distance indicator, $\eta_{n \triangleright m}$; by default, if the distances are specified, $\eta_{n \triangleright m} = 1/d_{n \triangleright m}$; otherwise, $\eta_{n \triangleright m} = 1$;
- the definition of cost indicator, $\varphi_{n \triangleright m}$; by default, if the costs are specified, $\varphi_{n \triangleright m} = 1/c_{n \triangleright m}$; otherwise, $\varphi_{n \triangleright m} = 1$;
- the importance weights: $\alpha \geq 0$, $\beta \geq 1$ and $\gamma \geq 1$; by default: $\alpha = \beta = \gamma = 1$;
- the pheromone evaporation factor, $\rho \in (0,1)$;
- the selection threshold of target state, $h_s \in [0,1]$.

➢ Stop test. For example, the maximum number of iterations to perform, $K_{\max} \in \mathbb{N}^*$ (even if there are some leaves in the graph).

2. Initialization.

a. Denote by $\mathcal{F} = \{f_p\}_{p \in \overline{1,P}}$ the ant colony. Choose at random or by a certain strategy the ants departure positions, $\mathcal{N} = \{n_0^f\}_{f \in \mathcal{F}} \subset \mathcal{S}$. Some positions could repeat, which means that several ants could start their journeys from the same node.

b. Initialize the memory of each ant with the departure point of its path (i.e. with the corresponding positions of $\mathcal{N}$). One assumes that the ant colony already is sorted by rank, as all the ants have the same performance, for now.

c. Initialize the set of ants that arrived on leaves of the graph (being thus blocked): $\mathcal{B} = \varnothing$.

d. Apply the amount $\tau_{\max}$ of pheromone on all arcs of search space. Store this value into the arcs memory.

e. Initialize the number of iterations: $k = 0$.

3. While $\mathcal{F} \neq \varnothing$ and $k \leq K_{\max}$, do:

3.1. For each ant $f \in \mathcal{F}$:

3.1.1. Determine the set of target states $\mathcal{E}_{n_k^f}(f)$ bond to state $n_k^f \in \mathcal{S}$, which currently is hosting the ant f. If the node has more than N_c children, use a U-PRSG to select the N_c required children.

3.1.2. If $\boldsymbol{\mathcal{E}}_{n_k^f}(f)=\varnothing$, , then the ant is blocked on a leaf. In this case, it is removed from the colony ($\mathcal{F} \leftarrow \mathcal{F} \setminus \{f\}$), being added to the set of blocked ants ($\mathcal{B} \leftarrow \mathcal{B} \cup \{f\}$).

3.1.3. Otherwise:

3.1.3.1. Evaluate the strategic criterion:

$$f_{n_k^f}[m]=\tau^{\alpha}_{n_k^f \triangleright m}\eta^{\beta}_{n_k^f \triangleright m}\varphi^{\gamma}_{n_k^f \triangleright m}, \quad \forall m \in \boldsymbol{\mathcal{E}}_{n_k^f}(f).$$

(In the beginning, $\tau_{n_0^f \triangleright m}=\tau_{\max}$, $\forall m \in \boldsymbol{\mathcal{E}}_{n_k^f}(f)$.)

3.1.3.2. Use a U-PRSG to select a number $h \in [0,1]$.

3.1.3.3. If $h \le h_s$, then the target state is determined by minimization of strategic criterion:

$$n_{k+1}^f = \underset{m \in \boldsymbol{\mathcal{E}}_{n_k^f}(f)}{\operatorname{argmin}} \left\{ f_{n_k^f}[m] \right\}.$$

3.1.3.4. Otherwise:

a. Construct the probability density of target states as suggested by equation [2.35]:

$$\wp_{n_k^f \triangleright m}=\frac{f_{n_k^f}[m]}{\sum_{n \in \boldsymbol{\mathcal{E}}_{n_k^f}(f)} f_{n_k^f}[n]}, \quad \forall m \in \boldsymbol{\mathcal{E}}_{n_k^f}(f).$$

b. With the resulted probability density, use a P-PRSG (based on BGA), to select the target state n_{k+1}^f. (Set sufficient resolution so that no children are missed.)

3.1.3.5. Move the ant f towards the target state n_{k+1}^f.

3.1.3.6. Evaluate the ant performance, $f_{n_0^f \triangleright \cdots \triangleright n_k^f \triangleright n_{k+1}^f}(f)$.

3.1.3.7. If the ant f is the first one that traveled the arc $n_k^f \triangleright n_{k+1}^f$, initialize $\mathcal{F}_{n_k^f \triangleright n_{k+1}^f}=\{f\}$. Otherwise, the ant will join the set of the other ants that traveled along the trail: $\mathcal{F}_{n_k^f \triangleright n_{k+1}^f} \leftarrow \mathcal{F}_{n_k^f \triangleright n_{k+1}^f} \cup \{f\}$.

3.1.3.8. Evaluate the amount of laid pheromone on the trail by the ant f, as suggested in equation [2.38]:

$$\Delta\tau_{n_k^f \triangleright n_{k+1}^f}(f)=\frac{Q}{f_{n_0^f \triangleright \cdots \triangleright n_k^f \triangleright n_{k+1}^f}(f)}.$$

3.2. If $\mathcal{F} \neq \varnothing$:

3.2.1. Sort by rang the ant colony $\mathcal{F}$. The best ants are taking the first positions: $\mathcal{F} = \{f_p\}_{p \in \overline{1,P}}$. Denote by $n_0^p \triangleright \cdots \triangleright n_k^p \triangleright n_{k+1}^p$ the trail of ant f_p ($p \in \overline{1,P}$).

3.2.2. For $p \in \overline{1,P}$, update the pheromone trace of trail $n_k^p \triangleright n_{k+1}^p$, as suggested by equation [2.37]:

$$\tau_{n_k^p \triangleright n_{k+1}^p} \leftarrow \min\left\{\tau_{\max}, (1-\rho)\tau_{n_k^p \triangleright n_{k+1}^p} + \sum_{f \in \mathcal{F}_{n_k^p \triangleright n_{k+1}^p}} \Delta\tau_{n_k^p \triangleright n_{k+1}^p}(f)\right\}.$$

3.2.3. For $r \in \overline{1,P_e}$, update the pheromone trace of (elite) path $n_0^r \triangleright \cdots \triangleright n_{k-1}^r \triangleright n_k^r$, as suggested by equations [2.46] and [2.47]. Compute first:

$$\Delta\tau^p_{n_{i-1}^r \triangleright n_i^r} = \begin{cases} \dfrac{Q}{f_{n_0^p \triangleright \cdots \triangleright n_{k-1}^p \triangleright n_k^p}(f_p)} & \text{, si } f_p \text{ a parcouru le trajet } n_{i-1}^r \triangleright n_i^r \\ 0 & \text{, sinon} \end{cases},$$

for each $i \in \overline{1,k}$ and $p \in \overline{1,P_e}$. Then, update the trace:

$$\tau_{n_{i-1}^r \triangleright n_i^r} \leftarrow \min\left\{\tau_{\max}, (1-\rho)\tau_{n_{i-1}^r \triangleright n_i^r} + \frac{\rho}{P_e}\sum_{p=1}^{P_e}(P_e - p + 1)\Delta\tau^p_{n_{i-1}^r \triangleright n_i^r}\right\},$$

for each $i \in \overline{1,k}$.

3.2.4. For all parent-child trial of type $n \triangleright m$ (but different from the trails of steps 3.2.2 and 3.2.3), simulate the evaporation phenomenon, as follows:

$$\tau_{n \triangleright m} \leftarrow \max\{(1-\rho)\tau_{n \triangleright m}, \tau_{\min}\}.$$

3.2.5. Evaluate the variance of the entire ant colony ($\mathcal{F} \cup \mathcal{B}$):

$$\sigma_{k+1} = \frac{1}{P}\sum_{f \in \mathcal{F} \cup \mathcal{B}}\left(f_{n_0 f \triangleright \cdots \triangleright n_k f \triangleright n_{k+1} f}(f) - \mu_{k+1}\right)^2,$$

where:

$$\mu_{k+1} = \frac{1}{P}\sum_{f \in \mathcal{F} \cup \mathcal{B}} f_{n_0 f \triangleright \cdots \triangleright n_k f \triangleright n_{k+1} f}(f).$$

3.2.6. If $k>1$ and $\frac{\sigma_{k+1}-\sigma_k}{\sigma_k}<\varepsilon$, the best ant has enough exploited its path and the search space exploration should be restarted through the non blocked ants of colony, in order to try other paths. To do so, the traces intensities corresponding to all parent-child trails, $n \triangleright m$, which do not belong to the elite, are increased, as suggested by equation [2.40]:

$$\tau_{n\triangleright m}\left(f_p\right) \leftarrow \lambda\tau_{n\triangleright m}\left(f_p\right)+(1-\lambda)\tau_{\max}, \quad \forall\, p \in \overline{P_e+1, P}\,.$$

3.2.7. Proceed with the next iteration: $k \leftarrow k+1$.

4. Choose the best ant in the set $\mathcal{F} \cup \mathcal{B}$.

5. Return:

- The optimal path, corresponding to the best ant.
- The optimal criterion value on the best ant path.

To increase the generality, the pheromone traces were updated by the ants ranking strategy. Also, the Max-Min technique was employed in the numerical procedure.

The Algorithm 2.6 is similar to a GA in terms of complexity and configuring difficulty. In turn, its efficiency substantially is higher than the one of Algorithm 2.5. In this procedure, beside the aforementioned improvements, a supplementary prevention measure has been taken. If the number of children per node is too large, the algorithm can become greedy. To avoid this risk, a maximum number of children per node has been enforced, namely N_e. As the children are selected at random, in turn, the algorithm might miss the global minimum and even important local minima. However, a satisfactory trade-off can be found by varying the number N_e.

The user can design various (and different) SACA, more ore less complex than the previous one, depending on the application. Nevertheless, if the exploration-exploitation trade-off is not well managed, the algorithm performance usually is modest. The most part of SACA design effort should focus on efficient control of this trade-off.

2.4.5. *Traveling salesman example*

Recall that, in this application, the problem is to find the shortest path passing once and only once through N cities. Consequently, the search space can easily be organized as a complete network, in which every node is assigned to a city and all the other nodes (cities) are its children. Obviously, all distances have to be known. The Figures 2.17 show an example of French cities network.

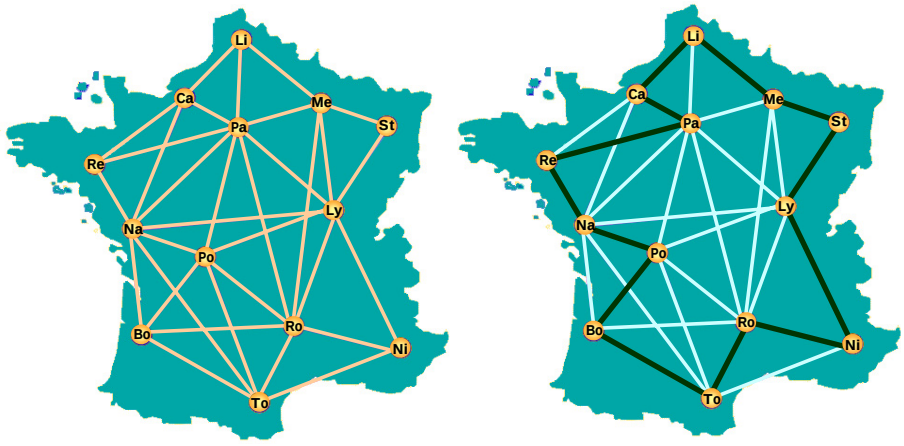


Figure 2.17. *Example of French cities network for the traveling salesmen problem: (a) in the beginning of search; (b) in the end of search.*

An ACA to solve this problem was introduced in [COL 92] and [DOR 92].

The corresponding ACA has the following peculiarity in this framework: each time an ant arrived into a city, the next target city can only be selected from the direct children that the ant has not visited yet (even though other ants might have already passed through those children). It follows that, for a node n hosting at least two ants, several children sets of type $\mathcal{E}_n$ can exist, depending on the ant of concern. Also, if the number of cities is too large, it is suitable to limit the size of sets $\mathcal{E}_n$. During the search, the number of target children gradually decreases down to the null value. This implies that the algorithm should run until every ant of the colony has reached a leaf (a node with no target children).

The Algorithm 2.7 constitutes an adaptation of ACA to the traveling salesman problem. Several differences from other ACA introduced into the literature can be noticed in this procedure.

Algorithm 2.7. *Optimization procedure by and colony, to solve the traveling salesman problem.*

1. Input data:

- Search space $\mathcal{S}$ including K cities to be visited by the traveling salesman. The space $\mathcal{S}$ is organized as an oriented graph with well defined distances for all the arcs between cities.
- Distance the salesman has traveled on the path of search space, f (criterion to minimize).

- Configuring parameters:
  - the number of ants in the colony, $P \in \mathbb{N}^*$;
  - the maximum number of target cities for each ant in the colony, $N_c \in \mathbb{N}^*$;
  - the initial amount of pheromone carried by each ant in the colony, $Q > 0$;
  - the pheromone trace intensity limits on trails from the search space, $0 < \tau_{\min} < \tau_{\max}$;
  - the correction factor of pheromone trace intensity, $\lambda \in [0,1]$;
  - the definition of distance indicator, $\eta_{n \triangleright m} = 1/d_{n \triangleright m}$;
  - the importance weights: $\alpha \geq 0$, $\beta \geq 1$ and $\gamma \geq 1$; by default: $\alpha = \beta = \gamma = 1$;
  - the pheromone evaporation factor, $\rho \in (0,1)$;.

1. Initialization.

a. Denote by $\mathcal{F} = \{f_p\}_{p \in \overline{1,P}}$ the ant colony. Choose at random or by a certain strategy the ants departure positions, $\mathcal{N} = \{n_0^f\}_{f \in \mathcal{F}} \subset \mathcal{S}$. The ants should start the search from P different cities, if possible.

b. Initialize the memory of each ant with the departure point of its path (i.e. with the corresponding positions of $\mathcal{N}$).

c. Apply the amount $\tau_{\max}$ of pheromone on all arcs of search space. Store this value into the arcs memory.

2. For $k \in \overline{1,K-1}$ (where k is the number of cities an ant has visited):

3.1. Initialize the set of all trails the ants were traveled along: $\mathcal{T} = \varnothing$. (By definition, such a trail only covers the arc between two cities.)

3.2. For each ant $f \in \mathcal{F}$:

3.2.1. Determine the set of target cities $\mathcal{E}_{n_k^f}(f)$ among the children of city $n_k^f \in \mathcal{S}$, which currently is hosting the ant f. This set is specific to every ant f. If the size of $\mathcal{E}_{n_k^f}(f)$ is bigger than N_c, use a U-PRSG to select the target cities. Note that $\mathcal{E}_{n_k^f}(f)$ cannot include the cities that the ant already has visited (i.e. $n_1^f, \cdots, n_k^f \notin \mathcal{E}_{n_k^f}(f)$).

3.2.2. Evaluate the strategic criterion:

$$f_{n_k^f}[m] = \tau_{n_k^f \triangleright m}^{\alpha} \eta_{n_k^f \triangleright m}^{\beta}, \quad \forall m \in \boldsymbol{\mathcal{E}}_{n_k^f}(f).$$

(In the beginning $\tau_{n_1^f \triangleright m} = \tau_{\max}$, $\forall m \in \boldsymbol{\mathcal{E}}_{n_k^f}(f)$.)

3.2.3. Construct the probability density of target cities, as suggested by equation [2.35]:

$$\wp_{n_k^f \triangleright m} = \frac{f_{n_k^f}[m]}{\sum_{n \in \boldsymbol{\mathcal{E}}_{n_k^f}(f)} f_{n_k^f}[n]}, \quad \forall m \in \boldsymbol{\mathcal{E}}_{n_k^f}(f).$$

3.2.4. With the resulted probability density, use a P-PRSG (based on BGA), to select the target city n_{k+1}^f. (Set sufficient resolution so that no possible target is missed.)

3.2.5. Move the ant f into the target city n_{k+1}^f.

3.2.6. Evaluate the ant performance, $f_{n_1^f \triangleright \cdots \triangleright n_k^f \triangleright n_{k+1}^f}(f)$.

3.2.7. If $n_k^f \triangleright n_{k+1}^f \notin \mathcal{T}$ (i.e. If the ant f e is the first one that traveled the arc $n_k^f \triangleright n_{k+1}^f$), update the trails set $\mathcal{T} \leftarrow \mathcal{T} \cup \{n_k^f \triangleright n_{k+1}^f\}$ and the set of ants that traveled this trail $\mathcal{F}_{n_k^f \triangleright n_{k+1}^f} = \{f\}$. Otherwise, the ant will join the set of the other ants that already traveled along the trail: $\mathcal{F}_{n_k^f \triangleright n_{k+1}^f} \leftarrow \mathcal{F}_{n_k^f \triangleright n_{k+1}^f} \cup \{f\}$, without changing the set $\mathcal{T}$.

3.2.8. Evaluate the amount of laid pheromone on the trail by the ant f, as suggested in equation [2.38]:

$$\Delta\tau_{n_k^f \triangleright n_{k+1}^f}(f) = \frac{Q}{f_{n_0^f \triangleright \cdots \triangleright n_k^f \triangleright n_{k+1}^f}(f)}.$$

3.3. For the arc $n \triangleright m$ from the cities network:

3.3.1. If $n \triangleright m \in \mathcal{T}$, update the pheromone trace, as suggested by equation [2.37]:

$$\tau_{n \triangleright m} \leftarrow \min\left\{\tau_{\max}, (1-\rho)\tau_{n \triangleright m} + \sum_{f \in \mathcal{F}_{n \triangleright m}} \Delta\tau_{n \triangleright m}(f)\right\}.$$

3.3.2. Otherwise, simulate the evaporation phenomenon as follows:

$$\tau_{n \triangleright m} \leftarrow \max\{(1-\rho)\tau_{n \triangleright m}, \tau_{\min}\}.$$

3. **Choose the best ant of $\mathcal{F}$ colony, i.e. the ant that traveled the shortest distance.**
4. **Return:**
   - The optimal sequence of cities to visit, corresponding to the path of the best selected ant.
   - The minimum distance on the best ant path.

If the number of cities to visit is reasonable small, it is not necessary to design a SACA as solving tool. Perhaps, an elitist strategy could increase the search speed of Algorithm 2.7. The user is free to try various versions of ACA or SACA in case of this problem.

The critical step of Algorithm 2.7 is 3.3, in which the pheromone traces of all network arcs have to be updated (by intensification or evaporation). This operation could be time consuming in case of big number of cities to visit and/or if the algorithm is not implemented on a parallel machine.

After running the ACA on the example of Figure 2.17(a), the solution of Figure 2.17(b) was obtained. In the beginning, the pheromone trace is constant for all the network arcs, as suggested in Figure 2.17(a). In the end, the Figure 2.17(b) shows that the pheromone trace is much stronger on the optimal path than on the other arcs.

Sometimes, the salesman is enforced to start the journey from a specific location. The corresponding city actually becomes a *graph root* and plays the role of colony nest. In this case, at the next iteration it is recommended to remove the limitation concerning the number of children (just for the root city) and, moreover, to make sure each ant is following a different arc, if possible. Nevertheless, if the departure is not specified, the Figure 2.17 shows a closed loop that can be started from any city.

2.5. Particle swarm optimization

2.5.1. *Basic metaheuristic*

2.5.1.1. *Principle*

This category of metaheuristics is based on a principle that already has been suggested by the ACA: during the search, there is a coordination at colony level, which, actually, expresses the swarm intelligence [BON 99], [EBE 01], [DRE 05]. To impress and maintain the pheromone trace along a trail, several ants are involved. This phenomenon reveals that the colony seems to have some *self conscience*, to which the most ants are contributing.

This idea was exploited in a direct manner within the framework of particle swarm algorithms, as firstly introduced in [KEN 95]. Here, instead of the ant

colony, a population of *particles* is managing the search. The population is guided according to some specific *cognitive conscience* of each particle and, at the same time, following some global *social conscience*. Beside the ants, some other animal swarms exhibited the existence of social conscience, namely the bees, the bats, and the fireflies. In this section, general algorithms concerning particle swarms optimal behavior are presented. In addition, particular algorithms simulating the behavior of some specific animal swarms are described as well.

The philosophy of *particle swarm optimization* (PSO) can easily be described as follows.

- In the beginning, each particle of the swarm is located at random in the search space, its speed also being selected at random.
- Every particle in the swarm is able to evaluate the quality of its own position and to keep in memory its best performance. All these together constitute an expression of its own *cognitive conscience*.
- At each search iteration, every particle can inquire some other particles (the *informants*) about their positions and best performances. In other words, the particle is becoming aware of the *social conscience* of the swarm (or, at least, of a part of it).
- At each search iteration too, every particle changes its position and speed, depending on the acquired information.
- In the end, the particles agglomerate a small vicinity around one (local or global) optimal point of preset criterion to optimize.

Figure 2.18 illustrates the principle of this optimization type, also known as *optimization by particles agglomeration*.

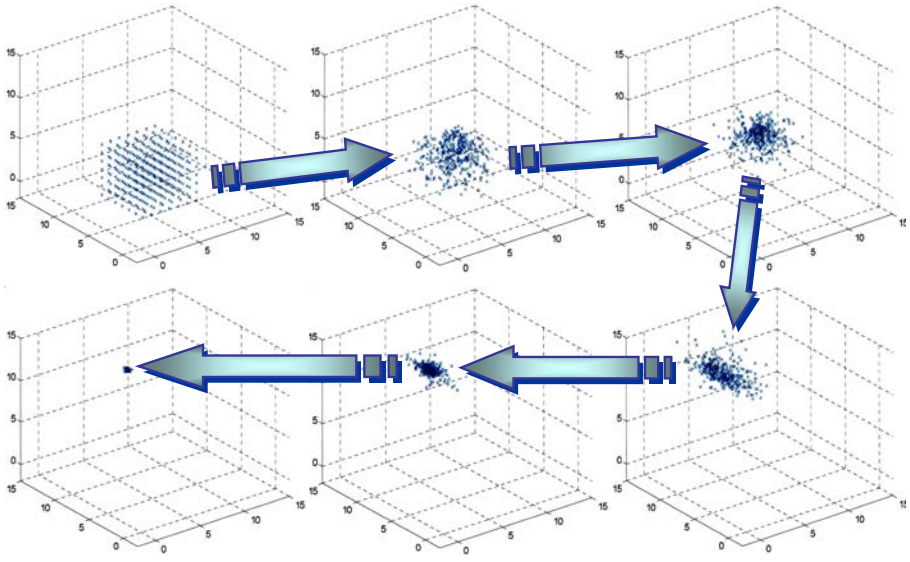


Figure 2.18. *Principle of particle swarm optimization.*

The corresponding algorithms are employed to solve the problem [1.2], for which the search space is not necessarily organized as a graph. In this framework, the search space rather is seen as a geographical zone to explore through dynamic particles, in order to find the optimum position of criterion [1.1].

2.5.1.2. *Particles dynamical model*

Consider the particle $p \in \overline{1,P}$ from the swarm (of size $P \in \mathbb{N}^*$) is located in position $\mathbf{x}_p^k \in \boldsymbol{\mathcal{S}}$ at iteration $k \in \mathbb{N}$. At the next iteration, the particle has to move into the position $\mathbf{x}_p^{k+1} \in \boldsymbol{\mathcal{S}}$. This movement is realizes along the route $\mathbf{x}_p^k \triangleright \mathbf{x}_p^{k+1}$, with constant speed $\mathbf{v}_p^k$, during the time delay ΔT_p^k. Then, naturally:

$$\mathbf{x}_p^{k+1} = \mathbf{x}_p^k + \mathbf{v}_p^k \Delta T_p^k, \quad \forall k \in \mathbb{N}. \qquad [2.49]$$

In this context, how to compute the speed $\mathbf{v}_p^k$ and the time delay ΔT_p^k becomes crucial. In general, the time delay either is constant (for example, $\Delta T_p^k = 1$, for each particle and iteration), or is selected at random, by means of a U-PRSG, in range $[0,T]$ (a priori known). Since the recursive equation [2.49] does not guarantee that the new particle position stays inside the search space $\boldsymbol{\mathcal{S}}$, the time delay ΔT_p^k can be employed to make the new position viable. For example, a technique to bring the new positions back into the search space is the following (by assuming the interval $[0,T]$ already is preset):

1. Compute the speed $\mathbf{v}_p^k$ (it will be shown later how).
2. Use a U-PRSG to select $\Delta T_p^k \in [0,T]$.
3. While $\mathbf{x}_p^k + \mathbf{v}_p^k \Delta T_p^k \notin \boldsymbol{\mathcal{S}}$, (repeatedly) use the U-PRSG to select a new ΔT_p^k, but in range from 0 to the current ΔT_p^k. More specifically, $\Delta T_p^k \leftarrow \text{GSPA-U}\left(\left[0, \Delta T_p^k\right]\right)$.

It can easily be noticed that, by this technique, the time delay gradually decreases, until the constraint $\mathbf{x}_p^k + \mathbf{v}_p^k \Delta T_p^k \in \boldsymbol{\mathcal{S}}$ is verified. In this case, the new particle position, $\mathbf{x}_p^{k+1}$, is given by $\mathbf{x}_p^k + \mathbf{v}_p^k \Delta T_p^k$, as computed in equation [2.49].

There are applications for which the search space only includes integer numbers. In this case, the equation [2.49] is replaced by:

$$\mathbf{x}_p^{k+1} = \left\lfloor \mathbf{x}_p^k + \mathbf{v}_p^k \Delta T_p^k + 0.5 \right\rfloor, \quad \forall k \in \mathbb{N}, \qquad [2.50]$$

where $\lfloor a \rfloor$ is the integer part of $a \in \mathbb{R}$, while $\lfloor a + 0.5 \rfloor$ returns the closest integer to the number $a \in \mathbb{R}$. The same technique of making positions viable applies here too, but the equation [2.50] is employed instead of equation [2.49].

Another technique of making positions viable is concerned with the components $\left\{ x_{p,i}^k \right\}_{i \in \overline{1,nx}}$ of vector $\mathbf{x}_p^k$, at iteration $k \in \mathbb{N}$. If the variation range of each component $x_{p,i}^k$ are known (say $x_{p,i}^{\min}$ and $x_{p,i}^{\max}$), then the following correction can be applied:

$$x_{p,i}^k \leftarrow \min\left\{ \max\left\{ x_{p,i}^k + v_{p,i}^k \Delta T_p^k, x_{p,i}^{\min} \right\}, x_{p,i}^{\max} \right\}. \qquad [2.51]$$

The advantages of making viable the positions by components are obvious. First, the technique solely applies on *rebel* components, that are violating the search space limits, and not to all components, as previously. Second, the new viable position seemingly is closer to the non viable one, which is less distorting the search strategy. If a component varies in a tight range, the time delay technique can produce severe modifications of the other components, with larger variation ranges.

There are some disadvantages too, when applying the technique [2.51]. Especially in case of non linear recursive equation [2.50], the particle might be stuck on the search space frontier. Or, in case of fractal optimization criteria, unlike the smooth criteria, it is very unlikely that the optima are located on the search space bounds. An important drawback of the technique [2.51] is that the user has to know in advance the variation ranges for all components. This requirement is not always easy to fulfill. For example, if the search space has spherical topology (and not a cubic one), it is quite difficult to specify the position variation ranges on components (as the limits of a component can depend on the values of the other components).

A third technique for making positions viable may result after combining the two previous techniques. This time, one randomly chosen time delay is associated to each speed component. Several random time delays are employed now, instead of a single one. By this technique, the new viable position cannot be trapped for longtime in the vicinity of current position or on the search space frontier. The other side of the coin is that the route cannot be traveled at constant speed (because the global time delay ΔT_p^k does not exist any more). But this is just a minor interpretation issue, which cannot really affect the PSO procedure.

So far, an answer was given to the question "Is it correct the definition of next position, as expressed in [2.49] or [2.50]?". Another question awaits for an answer though: "How to update the particle speed?".

In fact, computing the particle speed constitutes the kernel of this metaheuristic. In order to answer the question, one focuses on the informants selection method for each iteration, as the other particles can more or less determine the evolution of

current particle. In general, the informants selection strategy changes each time the best performance of the whole swarm was not improved. Once an optimal position being found, the speed is updated by observing that there are three tendencies the particle is tempted to follow (see Figure 2.19):

- an *adventurous* tendency, which means to continue the journey with the current speed;
- a *conservative* tendency, which means to go in direction of the best position the particle has currently found;
- a *panurgian* tendency, which means to blindly follow the direction towards the optimal point, as pointed by the informants.

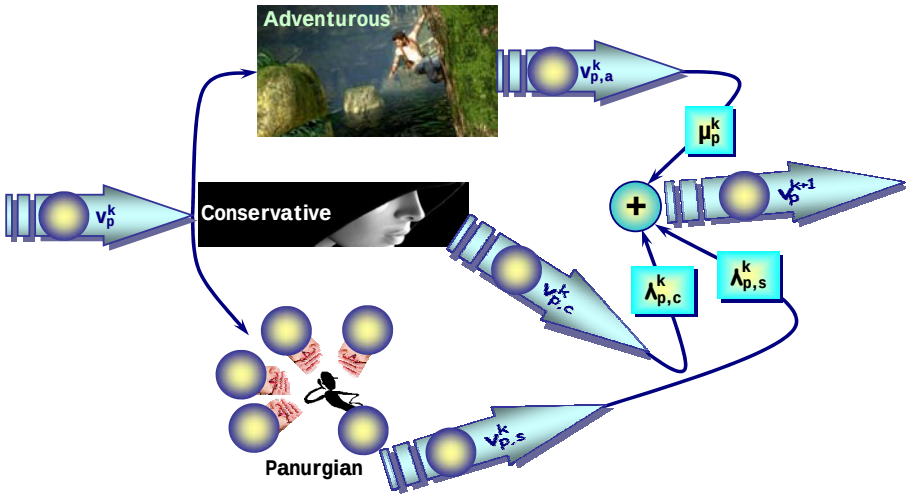


Figure 2.19. *Tendencies the particle is tempted to follow.*

The general equation of speed updating is then:

$$\mathbf{v}_p^{k+1} = \mu_p^k \mathbf{v}_p^k + \lambda_{p,c}^k \mathbf{v}_{p,c}^k + \lambda_{p,s}^k \mathbf{v}_{p,s}^k , \quad \forall\, p \in \overline{1,P} , \ \forall\, k \in \mathbb{N} , \qquad [2.52]$$

where: μ_p^k is a scalar referred to as *mobility factor* that usually depends on the particle inertia on its adventurous way;

$\lambda_{p,c}^k$ is a scalar expressing the *particle (own) conscience*, on its conservative way; this factor also is known as *cognitive variance* of particle;

$\mathbf{v}_{p,c}^k$ is the particle conservative vector speed towards its best current position;

$\lambda_{p,s}^k$ is a scalar expressing the *social conscience* of the swarm, which gives the particle reasons to follow a way pointed by the informants; this factor also is known as *social variance* of informants;

$\mathbf{v}_{p,s}^k$ is the panurgian vector speed towards the social optimum, as reported by informants.

In the simplest version of *PSO algorithm* (PSOA), the factors μ_p^k, $\lambda_{p,c}^k$ and $\lambda_{p,s}^k$ are constant. For example, μ_p^k varies in the interval $[0,1]$, a recommended value being $\mu_p^k = 0.72$, as reported in [KEN 95]. Also, $\lambda_{p,c}^k$ and $\lambda_{p,s}^k$ are equal to each other and vary in the interval $[0,2]$; the usual value is $\lambda_{p,c}^k = \lambda_{p,s}^k = 1.19$ (see the same reference).

The μ_p^k factor can control the exploration-exploitation trade-off. The smaller μ_p^k, the higher the convergence speed towards an optimum. In this case, the exploitation is enforced to detriment of exploration. On the contrary, when μ_p^k approaches the unit value, the space exploration dominates the search, which might slow down the convergence towards an optimum.

Another possible strategy is to select the three factors at random, by means of some U-PRSG (after tuning it on each variation interval). Perhaps the most efficient strategy is to adaptively update the factors, at each iteration and for each particle, depending on the dynamical characteristics of particles swarm. (Such characteristics are determined by the criterion to optimize and the search space bounds.) An adaptive version of PSOA is presented later on in this section.

Come back to equation [2.52]. Then the conservative speed is defined as follows:

$$\mathbf{v}_{p,c}^k = \frac{\mathbf{x}_p^{\text{opt},k} - \mathbf{x}_p^k}{\Delta T_{p,c}^k}, \quad \forall\, p \in \overline{1,P},\ \forall\, k \in \mathbb{N}, \qquad [2.53]$$

where $\mathbf{x}_p^{\text{opt},k}$ is the best position the particle has touched so far on its path (i.e. the position that led to the optimal value of criterion on the path), whilst $\Delta T_{p,c}^k$ is the time delay the particle needs to return to its best position $\mathbf{x}_p^{\text{opt},k}$, from the current position $\mathbf{x}_p^k$. Since the best particle position is known, the only problem with definition [2.53] is to find the time delay $\Delta T_{p,c}^k$. The simplest strategy is to set this variable at random in range $[0,T]$, by means of a U-PRSG. But more efficiently is to define $\Delta T_{p,c}^k$ adaptively, for example inversely proportional to the particle inertia (the more inertia, the longer the transition on the way back to the optimum).

The speed corresponding to panurgian tendency of equation [2.51] can be evaluated with the help of the following definition:

$$\mathbf{v}_{p,s}^{k} = \frac{\mathbf{x}_{\boldsymbol{\varepsilon}_p^k}^{\text{opt},k} - \mathbf{x}_p^m}{\Delta T_{p,s}^{k}}, \quad \forall\, p \in \overline{1,P}\,,\ \forall\, k \in \mathbb{N}\,, \tag{2.54}$$

where: $\boldsymbol{\varepsilon}_p^k$ is the group of informants that communicate with the particle of concern, $\mathbf{x}_{\boldsymbol{\varepsilon}_p^k}^{\text{opt},k}$ is the best current position, as communicated by the informants and $\Delta T_{p,s}^{k}$ is the time delay the particle needs to transit towards the best position $\mathbf{x}_{\boldsymbol{\varepsilon}_p^k}^{\text{opt},k}$, starting from the current position $\mathbf{x}_p^k$. The best position $\mathbf{x}_{\boldsymbol{\varepsilon}_p^k}^{\text{opt},k}$ is located on a path that one of the informants in the group already has traveled. Like in the previous case, the time delay $\Delta T_{p,s}^{k}$ is selected either at random (by means of a U-PRSG), or depending on the particle inertia.

The last problem to solve is to configure the informants group at each iteration. Some techniques to select the right informants are described next.

2.5.1.3. *Selecting the informants*

The informants group $\boldsymbol{\varepsilon}_p^k$ of definition [2.53] can be constituted by means of several techniques. The simpler way is to consider as informants all the particles in the swarm, but the current particle. In this case, a drawback is obvious: the time spent to query every informant can be too long, which may slow down the search, as too much exploration is performed. Of course, this exhaustive query process can be avoided by keeping in memory (and upgrading at each iteration) the best position of the entire particles swarm. In this case, only the most recent particle positions are involved into the updates, which increases the search speed.

Another technique is to define the size of an *elite* among the informants, say $P_i \leq P$, and to select the best position from the elite. Here, again, the elite has to be updated at each iteration, which could be time consuming. Moreover, the elite gradually begins to dominate the swarm, which puts too much weight on the exploitation.

Finally, in order to correctly balance the exploration-exploitation trade-off, the informants can be selected following the strategy below (at each iteration $k \in \mathbb{N}$ and for each particle $p \in \overline{1,P}$ of the swarm):

1. Conserve the best particle of swarm to be passed to the next informants group $\boldsymbol{\varepsilon}_p^{k+1}$ (its best position actually being of concern). This allows the user to avoid removing the global optimum, if found.
2. The other $P_i - 1$ positions of group $\boldsymbol{\varepsilon}_p^{k+1}$ are vacant and have to be taken by particles from the swarm, depending on the chance of each competitor and

on some activation threshold h_s, a priori known. Thus, while there still are vacant places in the $\mathcal{E}_p^{k+1}$ group, the following contest has to be organized:

a. For any particle of the whole swarm, selected at random, by means of a U-PRSG, pick a number $h \in [0,1]$, by means of a U-PRSG as well.
b. If $h \geq h_s$, then the particle is fortunate and takes the vacant position into the informants group $\mathcal{E}_p^{k+1}$.
c. Otherwise, the particle is refused and cannot become informant at this time. Nevertheless, the refused particle can be involved in a new contest, for another vacant place, at a future iteration.

Some other more or les sophisticated techniques to build the informants group can be designed. A technique based on evolutionary strategy is introduced in the sequel, as integrated in an adaptive version of PSOA.

2.5.2. *Standard PSO algorithm*

Usually, the particles start their movement from randomly selected positions in the search space. If there is knowledge about the existence of global optimum in a specific zone, then it is suitable to concentrate the search in that zone from the beginning. Otherwise, the particles can uniformly be distributed over the search space (as suggested by the first image of Figure 2.18). After setting the departure positions, the particles evolution should be started. A natural manner to make particles move is to randomly set the next position each particle has to reach for, together with the corresponding transition delays. The delays too are selected at random (but uniformly) in range $[0,T]$. The initial speed of each particle can thus be computed straightforwardly (as the ratio between the two positions difference and the transition delay).

To stop the search, very often, a maximum number of iterations is imposed, say $K_{\max} \in \mathbb{N}^*$. Another stop test concerns the survival of the best particle in the swarm. Thus, if the best particle is not overthrown by more than $M \in \mathbb{N}^*$ iterations, then it probably has touched an optimal point of the criterion.

The standard procedure of PSO is summarized by Algorithm 2.8.

Algorithm 2.8. *Standard procedure of PSO.*

1. Input data:

- Search space $\mathcal{S} \subseteq \mathbb{R}^{nx}$, seen as a geographical area where particles can move freely.
- Optimization criterion f (its definition) and type of optimum to search (minimum or maximum).

➢ Configuring parameters:

- the number of particles in the swarm, $P \in \mathbb{N}^*$ (a possible choice is: $P = 10 + 2\left\lfloor \sqrt{nx} \right\rfloor$); denote the swarm by $\boldsymbol{\mathcal{E}} \subseteq \boldsymbol{\mathcal{S}}$;
- the maximum number of informants, $P_i \geq 2$ (a possible choice is: $P_i \in 2, \overline{\left\lfloor \sqrt{\frac{P(P-1)}{2}} \right\rfloor}$);
- the maximum duration of any particle transition, $T > 0$;
- the mobility factor, $\mu \in [0,1]$ (by default: $\mu = 0.72$);
- the cognitive and social variances, λ_c, $\lambda_s \in [0,2]$ (by default: $\lambda_c = \lambda_s = 1.19$);
- the contest threshold to build the informants group, $h_s \in [0,1]$ (by default: $h_s = 0.75$);
- the maximum number of iterations, $K_{\max} \in \mathbb{N}^*$;
- the survival factor, $M \in \mathbb{N}^*$ (by default: $M = \left\lfloor 0.1 \cdot K_{\max} \right\rfloor$).

2. Initialization.

a. Distribute the P particles on the departure positions $\left\{\mathbf{x}_p^0\right\}_{p \in \overline{1,P}} \subset \boldsymbol{\mathcal{S}}$. (If no preferred technique exists, use a U-PRSG to uniformly select the positions from the search space.)

b. Use a U-PRSG to generate the initial speed vectors of particles, $\left\{\mathbf{v}_p^0\right\}_{p \in \overline{1,P}}$.

c. Initialize the best position and the best performance of each particle: $\left\{\left(\mathbf{x}_p^{\text{opt}} = \mathbf{x}_p^0, f_p^{\text{opt}} = f\left(\mathbf{x}_p^0\right)\right)\right\}_{p \in \overline{1,P}}$.

d. Detect the best particle of the swarm, i.e. solve the following optimization problem:

$$\underset{p \in \overline{1,P}}{\text{opt}} \left\{ f_p^{\text{opt}} \right\}$$

and store the solution $\left(\mathbf{x}_{\boldsymbol{\mathcal{E}},q}^{\text{opt}}, f_{\boldsymbol{\mathcal{E}},q}^{\text{opt}}\right)$ in memory, where $q \in \overline{1,P}$ is the swarm best particle index.

e. Initialize the survival index of current optimal particle: $m = 0$.

3. For $k \in \overline{1, K_{\max}}$ (where k is the iterations index) and while $m \le M$, do:

3.1. For each particle in the swarm, $p \in \overline{1, P}$:

3.1.1. Use a U-PRSG to select the particle transition delay: $\Delta T_p^{k-1} \subset [0, T]$.

3.1.2. Compute the position the particle can touch starting from the position $\mathbf{x}_p^{k-1}$, by means of equation [2.49] or [2.50]. For example:

$$\mathbf{x}_p^{k-1,k} = \mathbf{x}_p^{k-1} + \mathbf{v}_p^{k-1} \Delta T_p^{k-1}.$$

3.1.3. If necessary, make $\mathbf{x}_p^{k-1,k}$ viable, by a technique that can be adapted to the search space topology. (If possible, gradually decrease the rebel components of $\mathbf{x}_p^{k-1,k}$.) The current position of particle, $\mathbf{x}_p^k$, is thus obtained.

3.1.4. Compute the conservative speed of particle, with the help of definition [2.53]:

3.1.4.1. If $k = 1$, then $\mathbf{v}_{p,c}^0 = \mathbf{0}$.

3.1.4.2. Otherwise, set first the corresponding delay, $\Delta T_{p,c}^{k-1} \in [0, T]$ by using a U-PRSG and then evaluate:

$$\mathbf{v}_{p,c}^{k-1} = \frac{\mathbf{x}_p^{\text{opt}} - \mathbf{x}_p^{k-1}}{\Delta T_{p,c}^{k-1}}.$$

3.1.5. If $k = 1$, initialize the informants group: $\boldsymbol{\mathcal{E}}_p^0 = \{q\}$.

3.1.6. Otherwise, transfer the best informant of previous group, $\boldsymbol{\mathcal{E}}_p^{k-2}$, to next group, $\boldsymbol{\mathcal{E}}_p^{k-1}$.

3.1.7. For now, the informants group $\boldsymbol{\mathcal{E}}_p^{k-1}$ has a single particle. Therefore, the group $\boldsymbol{\mathcal{E}}_p^{k-1}$ has to be completed by contest. In the beginning, the competitor-particles group is identical to the global swarm $\boldsymbol{\mathcal{E}}$, excepting for the particle that already belongs to $\boldsymbol{\mathcal{E}}_p^{k-1}$. While the number of taken places in $\boldsymbol{\mathcal{E}}_p^{k-1}$ is smaller than P_i, do:

3.1.7.1. Use a U-PRSG to select a particle among the competitors.

3.1.7.2. Use a U-PRSG to select a number $h \in [0,1]$.

3.1.7.3. If $h \ge h_s$, then the particle won the contest, being selected as informant, and joins the group $\boldsymbol{\mathcal{E}}_p^{k-1}$.

3.1.7.4. Remove the winner from the competitors group.

3.1.7.5. If the competitors group is void, it can be set again by all particles of the swarm $\boldsymbol{\mathcal{E}}$ that do not belong to the group $\boldsymbol{\mathcal{E}}_p^{k-1}$.

3.1.8. Detect the best position as pointed by the informants, i.e. solve the following optimization problem:

$$\mathbf{x}_{\boldsymbol{\mathcal{E}}_p^{k-1}}^{\text{opt}} = \underset{r \in \boldsymbol{\mathcal{E}}_p^{k-1}}{\text{argopt}} \left\{ f_r^{\text{opt}} \right\}.$$

3.1.9. Compute the panurgian speed of particle with the help of definition [2.54]. Thus, set first the corresponding delay $\Delta T_{p,s}^{k-1} \in [0, T]$ by means of a U-PRSG and then evaluate:

$$\mathbf{v}_{p,s}^{k-1} = \frac{\mathbf{x}_{\boldsymbol{\mathcal{E}}_p^{k-1}}^{\text{opt}} - \mathbf{x}_p^{k-1}}{\Delta T_{p,s}^{k-1}}.$$

3.1.10. Evaluate the current particle speed with the help of equation [2.52]:

$$\mathbf{v}_p^k = \mu \mathbf{v}_p^{k-1} + \lambda_c \mathbf{v}_{p,c}^{k-1} + \lambda_s \mathbf{v}_{p,s}^{k-1}.$$

3.1.11. If the criterion value $f\left(\mathbf{x}_p^k\right)$ is better than f_p^{opt}, then $\mathbf{x}_p^{\text{opt}} = \mathbf{x}_p^k$ and $f_p^{\text{opt}} = f\left(\mathbf{x}_p^k\right)$. Otherwise, $\mathbf{x}_p^{\text{opt}}$ and f_p^{opt} are not changed.

3.2. Detect the best particle of swarm $\boldsymbol{\mathcal{E}}$, i.e. solve the following optimization problem:

$$\underset{p \in \overline{1,P}}{\text{opt}} \left\{ f_p^{\text{opt}} \right\}$$

and store the solution in memory.

3.3. If the resulted solution is identical to $\left(\mathbf{x}_{\boldsymbol{\mathcal{E}},q}^{\text{opt}}, f_{\boldsymbol{\mathcal{E}},q}^{\text{opt}}\right)$, increment the survival index: $m \leftarrow m+1$.

3.4. Otherwise:

3.4.1. Update the best current solution $\left(\mathbf{x}_{\boldsymbol{\mathcal{E}},q}^{\text{opt}}, f_{\boldsymbol{\mathcal{E}},q}^{\text{opt}}\right)$ by the result of problem in step 3.2. (The best particle index, q, can change or not.)

3.4.2. Reset the survival index: $m \leftarrow 0$.

4. Return:

- The best solution in the swarm: $\left(\mathbf{x}_{\boldsymbol{\mathcal{E}},q}^{\text{opt}}, f_{\boldsymbol{\mathcal{E}},q}^{\text{opt}}\right)$.

It has to be outlined that, within the Algorithm 2.8, every particle is endowed with a memory to store:

- the best touched position on the traveled path ($\mathbf{x}_p^{\text{opt}}$);
- the best performance, corresponding to the best position (f_p^{opt});
- the current position ($\mathbf{x}_p^k$);
- the previous position ($\mathbf{x}_p^{k-1}$).

Nevertheless, it is not necessary to keep all the particle path in memory.

As for the GA, the general convergence of *standard PSOA* (SPSOA) cannot be soundly proven. In the most applications, one can observe that, after a while, the particles agglomerate a narrow vicinity around some point of search space (which actually is the found optimal point), as the Figure 2.18 illustrates.

The user can control the exploration-exploitation trade-off through an indirect and rather empirical technique, by setting the μ and P_i parameters at will. In fact, the recommended values of those parameters (namely $\mu = 0.72$ and $P_i \in \overline{2, \left\lfloor \sqrt{\frac{P(P-1)}{2}} \right\rfloor}$) were obtained after testing similar procedures as the one of Algorithm 2.8 in various applications.

2.5.3. *Adaptive PSO algorithm with evolutionary strategy*

Directly managing the balance between the swarm diversity and its capacity to converge towards an optimum becomes a great necessity in complex applications. For this reason (and by necessity to implement the PSOA in such an application, of multi-variable phenomena and systems prediction, as presented in the end of sub-section), the main configuring parameters, namely μ, λ_c and λ_s cannot be set as constant and, moreover should not be determined empirically. Since the particle inertia, cognitive conscience and social conscience are natural concepts in PSO, it is suitable to quantify them in adaptive manner, depending on the swarm dynamics.

Let $p \in \overline{1,P}$ be the index of some arbitrarily chosen particle of swarm $\boldsymbol{\mathcal{E}}$ and consider $k \in \mathbb{N}$ as the index of current iteration.

In its adventuring attempt, the particle dynamics should naturally be weighted by a mobility factor, denoted by μ_p^k (see the equation [2.52] again). This factor opposes to the particle inertia, denoted by η_p^k. More specifically:

$$\mu_p^k = 1 - \eta_p^k, \quad \forall\, p \in \overline{1,P},\ \forall\, k \in \mathbb{N}. \qquad [2.55]$$

The farther the particle from its best position, the less inertia the particle should have and, thus, the more mobility. The particle is "motivated" to move faster, as being strongly attracted by the best position, in this case. On the contrary, the particle nearby its best position has no reason to go away and rather is tempted to stay "glued" on that zone. Consequently, the relative inertia of a particle can be defined as follows:

$$\eta_p^k = \frac{f\left(\mathbf{x}_p^k\right) - f\left(\mathbf{x}_p^{\text{tpo},k}\right)}{f\left(\mathbf{x}_p^{\text{opt},k}\right) - f\left(\mathbf{x}_p^{\text{tpo},k}\right)} \in [0,1], \quad \forall\, p \in \overline{1,P},\ \forall k \in \mathbb{N}, \qquad [2.56]$$

where, beside the known notations, $\mathbf{x}_p^{\text{tpo},k}$ stands for the worst position on the current path. The acronym "tpo" is obtained when reading the acronym "opt" inversely, from right to left, to suggest that the position is opposed to the optimal one. Practically, if $f\left(\mathbf{x}_p^{\text{opt},k}\right)$ is the best value of optimization criterion on particle current path, then $f\left(\mathbf{x}_p^{\text{tpo},k}\right)$ is the worst one.

The definition [2.56] leads to the wanted effect: the closer the particle to its best position (i.e. the closer $f\left(\mathbf{x}_p^k\right)$ to $f\left(\mathbf{x}_p^{\text{opt},k}\right)$), the closer the relative inertia to the unit value (notice the ratio in [2.56]). Consequently, the mobility factor [2.55] approaches the null value. In this case, the particle is very inert, not tempted by adventure, and moves very little from the current position. On the contrary, if $f\left(\mathbf{x}_p^k\right)$ is quite far away from the particle best performance, as the difference $f\left(\mathbf{x}_p^k\right) - f\left(\mathbf{x}_p^{\text{tpo},k}\right)$ is small, the relative inertia is nearly null, whilst the mobility factor approaches the unit value. In this case, the particle needs to rapidly leave its current position and is tempted to continue the adventure, in order to return to its best position or to find a better one.

There is no strong reason for which the cognitive variance of a particle, λ_c, equals the social variance of informants group, λ_s. Therefore, the two factors should vary independently. In this framework, the two variances are denoted like in equation [2.52], namely $\lambda_{p,c}^k$ and $\lambda_{p,s}^k$, respectively.

The cognitive variance can quantify how the particle stays close to or goes away from the path leading to the optimum. A possible definition is the following, by taking the relative value, just like in case of the inertia:

$$\lambda_{p,c}^k = 2\frac{\sigma_{p,c}^k - \sigma_{p,c}^{\min,k}}{\sigma_{p,c}^{\max,k} - \sigma_{p,c}^{\min,k}} \in [0,2], \quad \forall\, p \in \overline{1,P},\ \forall k \in \mathbb{N}, \qquad [2.57]$$

where: $\sigma_{p,c}^k$ is the absolute cognitive variance of the particle, whereas $\sigma_{p,c}^{\min,k}$ and $\sigma_{p,c}^{\max,k}$ are the minimum and the maximum values of this variance, respectively.

The absolute cognitive variance is defined as below:

$$\sigma_{p,c}^{k} = \frac{1}{k+1}\sum_{l=0}^{k}\left\|\mathbf{x}_p^l - \mathbf{x}_p^{\mathrm{opt},k}\right\|^2, \quad \forall\, p \in \overline{1,P},\ \forall\, k \in \mathbb{N}. \qquad [2.58]$$

By this definition, one quantifies the dispersion of particle successive positions on the current path, $\left\{\mathbf{x}_p^l\right\}_{l\in\overline{0,k}}$, with respect to the best current position, $\mathbf{x}_p^{\mathrm{opt},k}$. The difference between the maximum and minimum variance values in definition [2.57] quantifies if the particle is tempted to exploit de local zone where its best position lies (in case of small difference), or on the contrary, if the particle is driven to explore a larger zone (in case of great difference).

Several scenarios are encompassed by the definition [2.57] between two situations at the limit. The particle is strongly conservative when $\lambda_{p,c}^{k}$ is big, i.e. when its cognitive absolute variance [2.58] is close to the maximum value, $\sigma_{p,c}^{\max,k}$. This involves the conservative tendency is strengthened in case the paths are quite scattered around the particle best position. In turn, the conservative tendency strongly is attenuated if the particle path stays concentrated around the best position, in which case the current absolute variance approaches its minimum, $\sigma_{p,c}^{\min,k}$.

From the implementation point of view, the definition [2.58] has a caveat: the entire path of every particle has to be kept in memory and updated, at each iteration. Fortunately, if the best position does not change after the transition $\mathbf{x}_p^{k-1} \triangleright \mathbf{x}_p^{k}$, i.e. if $\mathbf{x}_p^{\mathrm{opt},k-1} = \mathbf{x}_p^{\mathrm{opt},k}$ (and, thus, $\mathbf{x}_p^{k} \neq \mathbf{x}_p^{\mathrm{opt},k}$), then an interesting recursive equation of absolute variance can be written:

$$\sigma_{p,c}^{k} = \frac{k}{k+1}\sigma_{p,c}^{k-1} + \frac{1}{k+1}\left\|\mathbf{x}_p^{k} - \mathbf{x}_p^{\mathrm{opt},k-1}\right\|^2, \quad \forall\, p \in \overline{1,P},\ \forall\, k \in \mathbb{N}. \qquad [2.59]$$

In the beginning of search, the algorithm runs in a transient phase and the best position can change quite rapidly, from iteration to iteration. After the transient phase, the best position changes quite seldom, once for several successive iterations. The recursive equation [2.59] can thus alleviate the computational burden, even though the entire particle path still has to be kept in memory (and updated), until the search is completed.

The relative social variance $\lambda_{p,s}^{k}$ expresses the intensity of panurgian tendency the particle is tempted to follow, depending on how the informants group $\boldsymbol{\mathcal{E}}_p^k$ is concentrated or scattered around the best pointed position, $\mathbf{x}_{\boldsymbol{\mathcal{E}}_p^k}^{\mathrm{opt},k}$. Normally, if the informants are quite tightly grouped around their best position, the particle becomes quite confident in the information about this position and is temped to increase its

panurgian speed towards it. Otherwise, the particle should have doubts about the communicated best position and, by caution, is limiting its panurgian speed. Consequently, the relative social variance in defined as follows:

$$\lambda_{p,s}^{k} = 2\frac{\sigma_{p,s}^{\max,k} - \sigma_{p,s}^{k}}{\sigma_{p,s}^{\max,k} - \sigma_{p,s}^{\min,k}} \in [0,2], \quad \forall\, p \in \overline{1,P},\ \forall\, k \in \mathbb{N}, \tag{2.60}$$

where: $\sigma_{p,s}^{k}$ is the absolute social variance of informants group, whereas $\sigma_{p,s}^{\min,k}$ and $\sigma_{p,s}^{\max,k}$ are the minimum and the maximum values of this variance, respectively.

The absolute social variance of informants is defined below:

$$\sigma_{p,s}^{k} = \frac{1}{P_i}\sum_{r=1}^{P_i}\left\|\mathbf{x}_r^k - \mathbf{x}_{\boldsymbol{\varepsilon}_p^k}^{\text{opt},k}\right\|^2, \quad \forall\, p \in \overline{1,P},\ \forall\, k \in \mathbb{N} \tag{2.61}$$

and quantifies how the informants are dispersed around their best position. Differently from the absolute cognitive variance, in definition [2.61], only the informants current positions are taken into account, since solely the actual dispersion is of concern (and not the obsolete ones).

With the above definitions of factors μ_p^k, $\lambda_{p,c}^k$ and $\lambda_{p,s}^k$, the speed [2.52] is computed now adaptively. Although the three factors involve a good control of exploration-exploitation trade-off within the algorithm, a direct control is desirable, like in case of GA. This type of control can be achieved by means of an evolutionary strategy.

The global particles swarm, $\boldsymbol{\varepsilon}$, evolves towards an optimal point as suggested in Figure 2.18. But, during this evolution, contrary to what the figure displays, the dispersion of swarm can vary in various manners and even it can oscillate. (In the figure, the swarm dispersion just decreases, as the particles gradually are grouping themselves around a point.)

The swarm variance can control its diversity and is defined as follows:

$$\sigma_{\boldsymbol{\varepsilon}}^{k} = \frac{1}{P}\sum_{p=1}^{P}\left\|\mathbf{x}_p^k - \mathbf{x}_{\boldsymbol{\varepsilon},q}^{\text{opt},k}\right\|^2, \quad \forall\, k \in \mathbb{N}, \tag{2.62}$$

where $\mathbf{x}_{\boldsymbol{\varepsilon},q}^{\text{opt},k}$ is the current optimal position of the entire particles swarm, whilst $q \in \overline{1,P}$ is the index of particle that touched this best position. (It is not necessary that the optimal position $\mathbf{x}_{\boldsymbol{\varepsilon},q}^{\text{opt},k}$ belongs to the set of current positions of swarm particles, $\left\{\mathbf{x}_p^k\right\}_{p\in\overline{1,P}}$.)

The variance [2.62] can be monitored, in order to detect the swarm tendencies in terms of dispersion around the current optimal position. If, on one side, the variance

significantly increases in a few iterations, then the swarm tends to become too scattered and the exploration begins to dominate the exploitation. On the other side, if the variance sensibly decreases in a few iterations, then the swarm can be trapped into some narrow vicinity of its current optimal point, which is not necessarily the global one. In this case, the exploitation dominates the exploration.

To avoid the extreme situations above, an alarm is send each time the swarm relative variance suddenly changes (up or down), in just a few iterations. For example, if the relative variation of this variance is larger than the threshold $\alpha \in (0,1]$, or smaller than $-\alpha$, the alarm is enabled. The relative variation can be computed straightforwardly:

$$\rho_{\boldsymbol{\varepsilon}}^{k} = \frac{\sigma_{\boldsymbol{\varepsilon}}^{k} - \sigma_{\boldsymbol{\varepsilon}}^{k-1}}{\sigma_{\boldsymbol{\varepsilon}}^{k-1}}, \quad \forall k \in \mathbb{N}^{*}. \qquad [2.63]$$

For example, if $\rho_{\boldsymbol{\varepsilon}}^{k} > \alpha = 0.25$, then the swarm became 25% too scattered and its diversity should be decreased, in order to prevent oscillations in the convergence process. If, on the contrary, $\rho_{\boldsymbol{\varepsilon}}^{k} < -\alpha = -0.25$, then the swarm became 25% too focused on the current optimal point and its diversity should be increased, in order for the swarm to escape from this possible trap.

How to increase/decrease the swarm diversity in a direct manner? Partially and indirectly, this diversity is influenced by the informants. Even in case the user can control the informants selection, he/she will never control the entire population diversity in direct manner. To reach for this goal, a promising approach is to employ an evolutionary strategy. Such an approach is described next.

Beside the swarm, two other particle groups can evolve in parallel: one group referred to as *elite*, denoted by $\boldsymbol{\mathcal{E}}^{k}$, and another group referred to as *crowd*, denoted by $\boldsymbol{\mathcal{W}}^{k}$. The elite includes the best positions, as touched by the particles of swarm during their journeys, together with the corresponding speeds: $\boldsymbol{\mathcal{E}}^{k} = \left\{\left(\mathbf{x}_{p}^{\text{opt},k}, \mathbf{v}_{p}^{\text{opt},k}\right)\right\}_{p\in\overline{1,P}}$. On the contrary, the crowd includes the worst particle positions, together with the corresponding speeds: $\boldsymbol{\mathcal{W}}^{k} = \left\{\left(\mathbf{x}_{p}^{\text{tpo},k}, \mathbf{v}_{p}^{\text{tpo},k}\right)\right\}_{p\in\overline{1,P}}$.

In this context, the following strategy can be employed to control the swarm diversity:

1. Apply ranking of elite particles in $\boldsymbol{\mathcal{E}}^{k}$, depending on their performance (given by the optimization criterion), for example in decreasing order of the performance. The ranking is passed to the swarm $\boldsymbol{\mathcal{E}}$ too.
2. If the swarm $\boldsymbol{\mathcal{E}}$ became too concentrated (or less diverse), then:
   a. Split the swarm into three groups (not necessarily equal as number of particles to include): a first one corresponding to the elite high rank particles (to preserve in their current positions), a second one including

the particles to be replaced by resulted offspring, after applying crossovers between some other particles and a third one with particles to be replaced at random by newly created particles with initial locations outside the swarm.

b. Apply crossovers between particles of swarm $\mathcal{E}$ and particles of crowd $\mathcal{W}^k$, in order to take the vacant positions in the second group above. (By crossover, not only new positions are generated for the offspring, by also new speed vectors corresponding to those positions.) Each vacant place will be taken by the best particle among the two parents and two offspring. (One of the parents can thus conserve its place in the swarm or it can be replaced by another particle.)

c. Select at random (by means of a U-PRSG) the new positions and speeds to fill in the third group. The newly generated positions have to be different from any position the swarm particles have touched so far.

3. If the swarm $\mathcal{E}$ became too scattered (or more diverse), then:

a. Split the swarm into two groups (not necessarily equal as number of particles to include): a first one corresponding to the elite high rank particles (to preserve in their current positions) and a second one including the particles to be replaced by resulted offspring, after applying crossovers between some other particles.

b. Apply crossovers between particles of swarm $\mathcal{E}$ and particles of elite $\mathcal{E}^k$, in order to take the vacant positions in the second group above. Each vacant place will be taken by the best particle among the two parents and two offspring.

4. For each new particle in the swarm $\mathcal{E}$, the history of its evolution is void, as the particle just has been born.

5. Update the elite $\mathcal{E}^k$ and the crowd $\mathcal{W}^k$, by accounting the fact that, for each new particle in the swarm its initial position is both the best and the worst.

After the employment of strategy above, the swarm can evolve normally regardless the evolution history each particle may have or not.

The crossover between two particles can be performed following a natural rule. To be more specific, consider two particles in positions $\mathbf{x}_1$ and $\mathbf{x}_2$, respectively, as well as their corresponding speeds $\mathbf{v}_1$ and $\mathbf{v}_2$. Then, after applying crossover between them, two offspring are resulting, in positions:

$$\mathbf{y}_1 = \gamma \cdot \mathbf{x}_1 + (1-\gamma) \cdot \mathbf{x}_2 \quad \text{and} \quad \mathbf{y}_2 = \gamma \cdot \mathbf{x}_2 + (1-\gamma) \cdot \mathbf{x}_1 , \qquad [2.64]$$

while the corresponding speeds are:

$$\mathbf{w}_1 = \gamma \cdot \mathbf{v}_1 + (1-\gamma) \cdot \mathbf{v}_2 \quad \text{and} \quad \mathbf{w}_2 = \gamma \cdot \mathbf{v}_2 + (1-\gamma) \cdot \mathbf{v}_1 . \qquad [2.65]$$

In definitions [2.64] and [2.65], the γ parameter expresses the probability for the two parents to meet and to be committed to the crossover. Normally, this probability depends on the angle between the speeds of the two parents:

$$\gamma = \frac{1}{2} - \frac{\langle \mathbf{v}_1, \mathbf{v}_2 \rangle}{2\|\mathbf{v}_1\|\|\mathbf{v}_2\|} \in [0,1]. \qquad [2.66]$$

In definition [2.66], $\langle \mathbf{v}_1, \mathbf{v}_2 \rangle$ stands for the scalar product between vectors $\mathbf{v}_1$ and $\mathbf{v}_2$, whereas the ratio between this scalar product and the two vectors norm is the cosine of their angle. As the cosine varies in range from -1 to $+1$, it follows the probability γ varies from 0 to $+1$, which proves the definition [2.66] is correct.

Naturally, the mating probability is maximum if the parents are moving in opposite directions (when the cosine is minimum) and decreases as long as their speeds become collinear in the same direction (when the cosine is maximum).

The optimization procedure including the approach above is summarized within the Algorithm 2.9.

Algorithm 2.9. *Adaptive procedure of PSO, with evolutionary strategy.*

1. Input data:

- Search space $\mathcal{S} \subseteq \mathbb{R}^{nx}$, seen as a geographical area where particles can move freely.
- Optimization criterion f (its definition) and type of optimum to search (minimum or maximum).
- Configuring parameters:
  - the number of particles in the swarm, $P \in \mathbb{N}^*$ (a possible choice is: $P = 10 + 2\lfloor \sqrt{nx} \rfloor$); let $\mathcal{E} \subseteq \mathcal{S}$ denote the swarm;
  - the maximum number of informants, $P_i \geq 2$ (a possible choice is: $P_i \in 2, \overline{\left\lfloor \sqrt{\frac{P(P-1)}{2}} \right\rfloor}$);
  - the number of the best particles to preserve into the swarm at each iteration, $P_e \geq 1$ (by default: $P_e = \lfloor 0.4 \cdot P \rfloor$);
  - the number of particles to replace by applying crossover between other particles, if the swarm suddenly becomes too concentrated, $P_c \geq 1$ (by default: $P_c = \lfloor P/2 \rfloor$);
  - the maximum duration of any particle transition, $T > 0$;
  - the mobility factor in the transient phase, $\mu \in [0,1]$ (by default: $\mu = 0.72$);

- the relative cognitive and relative variances in the transient phase, $\lambda_c,\ \lambda_s \in [0,2]$ (by default: $\lambda_c = \lambda_s = 1.19$);
- the contest threshold to build the informants group, $h_s \in [0,1]$ (by default: $h_s = 0.75$);
- the alarm threshold to control the exploration-exploitation trade-off, $\alpha \in [0,1]$ (by default: $\alpha = 0.25$);
- the maximum number of iterations, $K_{\max} \in \mathbb{N}^*$;
- the minimum size of particle evolution history, before commuting to the adaptive strategy in the procedure, $K_{\min} \geq 2$ (by default, $K_{\min} = \max\{5\ ,\ 0.05 \cdot K_{\max}\}$) ;
- l the survival factor, $M \in \mathbb{N}^*$
  (by default: $M = \max\{K_{\min}, \lfloor 0.1 \cdot K_{\max} \rfloor\}$).

2. Initialization

a. Distribute the P particles on the departure positions $\left\{\mathbf{x}_p^0\right\}_{p\in\overline{1,P}} \subset \mathcal{S}$. (If no preferred technique exists, use a U-PRSG to uniformly select the positions from the search space.)

b. Use a U-PRSG to generate the initial speed vectors of particles, $\left\{\mathbf{v}_p^0\right\}_{p\in\overline{1,P}}$.

c. Denote by $\mathcal{E}^0$ the set of couples position-speed $\left(\mathbf{x}_p^0, \mathbf{v}_p^0\right)$ for all particles, $p \in \overline{1,P}$. By convention, $\mathcal{E}^k$ is the generation of $\mathcal{E}$ swarm at iteration $k \in \mathbb{N}$.

d. Initialize the best position and the best performance of each particle: $\left\{\left(\mathbf{x}_p^{\text{opt}} = \mathbf{x}_p^0,\ f_p^{\text{opt}} = f\left(\mathbf{x}_p^0\right)\right)\right\}_{p\in\overline{1,P}}$. These couples also serve as the worst position and performance of particles: $\left\{\left(\mathbf{x}_p^{\text{tpo}} = \mathbf{x}_p^0,\ f_p^{\text{tpo}} = f\left(\mathbf{x}_p^0\right)\right)\right\}_{p\in\overline{1,P}}$

e. Initialize the history index for each particle in the swarm: $k_p = 0$, $\forall\ p \in \overline{1,P}$.

f. Detect the best particle of the swarm, i.e. solve the following optimization problem:

$$\underset{p\in\overline{1,P}}{\text{opt}} \left\{ f_p^{\text{opt}} \right\}$$

and store the solution $\left(\mathbf{x}_{\mathcal{E},q}^{\text{opt}}, f_{\mathcal{E},q}^{\text{opt}}\right)$ in memory, where $q \in \overline{1,P}$ is the swarm best particle index.

g. Set the initial elite group: $\mathcal{B}^0 = \mathcal{E}^0$.

h. Set the initial crowd group: $\mathcal{W}^0 = \mathcal{E}^0$.

i. Initialize the survival index of current optimal particle: $m = 0$.

3. For $k \in \overline{1, K_{\max}}$ (where k is the iterations index) and while $m \leq M$, do:

3.1. Initialize the current generation: $\mathcal{E}^k = \varnothing$.

3.2. For each particle in the swarm, $p \in \overline{1, P}$:

3.2.1. Use a U-PRSG to select the particle transition delay: $\Delta T_p^{k-1} \subset [0, T]$.

3.2.2. Compute the position the particle can touch starting from the position $\mathbf{x}_p^{k-1}$, by means of equation [2.49] or [2.50]. For example:

$$\mathbf{x}_p^{k-1,k} = \mathbf{x}_p^{k-1} + \mathbf{v}_p^{k-1} \Delta T_p^{k-1}.$$

3.2.3. If necessary, make $\mathbf{x}_p^{k-1,k}$ viable, by a technique that can be adapted to the search space topology. (If possible, gradually decrease the rebel components of $\mathbf{x}_p^{k-1,k}$.) The current position of particle, $\mathbf{x}_p^k$, is thus obtained.

3.2.4. If $k > k_p + K_{\min}$, adaptively compute the mobility factor, by using the definitions [2.55] and [2.56]:

$$\mu_p^{k-1} = 1 - \frac{f\left(\mathbf{x}_p^{k-1}\right) - f\left(\mathbf{x}_p^{\text{tpo}}\right)}{f\left(\mathbf{x}_p^{\text{opt}}\right) - f\left(\mathbf{x}_p^{\text{tpo}}\right)}.$$

3.2.5. Otherwise, keep unchanged the mobility factor: $\mu_p^{k-1} = \mu$.

3.2.6. If $k > k_p + K_{\min}$, adaptively compute the relative cognitive variance, by using the definition [2.57]:

$$\lambda_{p,c}^{k-1} = 2 \frac{\sigma_{p,c}^{k-1} - \sigma_{p,c}^{\min}}{\sigma_{p,c}^{\max} - \sigma_{p,c}^{\min}}.$$

3.2.7. Otherwise, keep unchanged the relative cognitive variance: $\lambda_{p,c}^{k-1} = \lambda_c$.

3.2.8. Compute the conservative speed of particle, by using the definition [2.53]. To do so, set first the corresponding transition delay, $\Delta T_{p,c}^{k-1} \in [0, T]$, by means of a U-PRSG, and then evaluate: $\mathbf{v}_{p,c}^{k-1} = \dfrac{\mathbf{x}_p^{\text{opt}} - \mathbf{x}_p^{k-1}}{\Delta T_{p,c}^{k-1}}$.

3.2.9. If $k > k_p + 1$, transfer the best informant from the previous group, $\boldsymbol{\mathcal{E}}_p^{k-2}$, to the next group, $\boldsymbol{\mathcal{E}}_p^{k-1}$.

3.2.10. Otherwise, initialize $\boldsymbol{\mathcal{E}}_p^{k-1}$ by $\{q\}$.

3.2.11. For now, the informants group $\boldsymbol{\mathcal{E}}_p^{k-1}$ has a single particle. Therefore, the group $\boldsymbol{\mathcal{E}}_p^{k-1}$ has to be completed by contest. In the beginning, the competitor-particles group is identical to the global swarm $\boldsymbol{\mathcal{E}}$, excepting for the particle that already belongs to $\boldsymbol{\mathcal{E}}_p^{k-1}$. While the number of taken places in $\boldsymbol{\mathcal{E}}_p^{k-1}$ is smaller than P_i, do:

3.2.11.1. Use a U-PRSG to select a particle among the competitors.

3.2.11.2. Use a U-PRSG to select a number $h \in [0,1]$.

3.2.11.3. If $h \geq h_s$, then the particle won the contest, being selected as informant, and joins the group $\boldsymbol{\mathcal{E}}_p^{k-1}$.

3.2.11.4. Remove the winner from the competitors group.

3.2.11.5. If the competitors group is void, it can be set again by all particles of the swarm $\boldsymbol{\mathcal{E}}$ that do not belong to the group $\boldsymbol{\mathcal{E}}_p^{k-1}$.

3.2.12. Detect the best position as pointed by the informants, i.e. solve the following optimization problem:

$$\mathbf{x}_{\boldsymbol{\mathcal{E}}_p^{k-1}}^{\text{opt}} = \underset{r \in \boldsymbol{\mathcal{E}}_p^{k-1}}{\text{argopt}} \left\{ f_r^{\text{opt}} \right\}.$$

3.2.13. If $k > k_p + K_{\min}$, adaptively compute the relative social variance, by using the definition [2.60]:

$$\lambda_{p,s}^{k-1} = 2 \frac{\sigma_{p,s}^{\max} - \sigma_{p,s}^{k-1}}{\sigma_{p,s}^{\max} - \sigma_{p,s}^{\min}}.$$

3.2.14. Otherwise, keep unchanged the relative social variance: $\lambda_{p,s}^{k-1} = \lambda_s$.

3.2.15. Compute the panurgian speed of particle with the help of definition [2.54]. Thus, set first the corresponding delay $\Delta T_{p,s}^{k-1} \in [0, T]$ by means of a U-PRSG and then evaluate:

$$\mathbf{v}_{p,s}^{k-1} = \frac{\mathbf{x}_{\boldsymbol{\mathcal{E}}_p^{k-1}}^{\text{opt}} - \mathbf{x}_p^{k-1}}{\Delta T_{p,s}^{k-1}}.$$

3.2.16. Evaluate the current particle speed with the help of equation [2.52]:

$$\mathbf{v}_p^k = \mu_p^{k-1}\mathbf{v}_p^{k-1} + \lambda_{p,c}^{k-1}\mathbf{v}_{p,c}^{k-1} + \lambda_{p,s}^{k-1}\mathbf{v}_{p,s}^{k-1} .$$

3.2.17. Update the current generation: $\boldsymbol{\mathcal{E}}^k \leftarrow \boldsymbol{\mathcal{E}}^k \cup \left\{\left(\mathbf{x}_p^k, \mathbf{v}_p^k\right)\right\}$.

3.2.18. Update the particle evolution history:

3.2.18.1. Add the current position $\mathbf{x}_p^k$ to the current path.

3.2.18.2. If the criterion value $f\left(\mathbf{x}_p^k\right)$ is better than f_p^{opt}, then $\mathbf{x}_p^{\text{opt}} = \mathbf{x}_p^k$ and $f_p^{\text{opt}} = f\left(\mathbf{x}_p^k\right)$. Otherwise, $\mathbf{x}_p^{\text{opt}}$ and f_p^{opt} are not changed.

3.2.18.3. If the criterion value $f\left(\mathbf{x}_p^k\right)$ is worst than f_p^{tpo}, then $\mathbf{x}_p^{\text{tpo}} = \mathbf{x}_p^k$ and $f_p^{\text{tpo}} = f\left(\mathbf{x}_p^k\right)$. Otherwise, $\mathbf{x}_p^{\text{tpo}}$ and f_p^{tpo} are not changed.

3.2.18.4. If $k > k_p + 1$:

a. Evaluate the absolute cognitive variance by means of definition [2.58] (or, if possible, of definition [2.59]):

$$\sigma_{p,c}^k = \frac{1}{k - k_p + 1}\sum_{l=k_p}^{k}\left\|\mathbf{x}_p^l - \mathbf{x}_p^{\text{opt}}\right\|^2 .$$

b. Update the extremes of absolute cognitive variance:

$$\begin{cases}\sigma_{p,c}^{\min} \leftarrow \min\left\{\sigma_{p,c}^{\min}, \sigma_{p,c}^k\right\} \\ \sigma_{p,c}^{\max} \leftarrow \max\left\{\sigma_{p,c}^{\max}, \sigma_{p,c}^k\right\}\end{cases} .$$

c. Evaluate the absolute social variance by means of definition [2.61]:

$$\sigma_{p,s}^k = \frac{1}{P_i}\sum_{r=1}^{P_i}\left\|\mathbf{x}_r^k - \mathbf{x}_{\boldsymbol{\mathcal{E}}_p^k}^{\text{opt}}\right\|^2 .$$

d. Update the extremes of absolute social variance:

$$\begin{cases}\sigma_{p,s}^{\min} \leftarrow \min\left\{\sigma_{p,s}^{\min}, \sigma_{p,s}^k\right\} \\ \sigma_{p,s}^{\max} \leftarrow \max\left\{\sigma_{p,s}^{\max}, \sigma_{p,s}^k\right\}\end{cases} .$$

3.2.18.5. Otherwise:

a. Initialize the extremes of absolute cognitive variance: $\sigma_{p,c}^{\min} = \sigma_{p,c}^{\max} = \left\| \mathbf{x}_p^k - \mathbf{x}_p^{k-1} \right\|$.

b. Initialize the extremes of absolute social variance: $\sigma_{p,s}^{\min} = \sigma_{p,s}^{\max} = \frac{1}{P_i} \sum_{r=1}^{P_i} \left\| \mathbf{x}_r^k - \mathbf{x}_{\boldsymbol{\varepsilon}_p^k}^{\text{opt}} \right\|^2$.

3.3. Detect the best particle of swarm $\boldsymbol{\mathcal{E}}$, i.e. solve the following optimization problem:

$$\underset{p \in \overline{1,P}}{\text{opt}} \left\{ f_p^{\text{opt}} \right\}$$

and store the solution in memory.

3.4. If the resulted solution is identical to $\left(\mathbf{x}_{\boldsymbol{\varepsilon},q}^{\text{opt}}, f_{\boldsymbol{\varepsilon},q}^{\text{opt}} \right)$, increment the survival index: $m \leftarrow m+1$.

3.5. Otherwise:

3.5.1. Update the best current solution $\left(\mathbf{x}_{\boldsymbol{\varepsilon},q}^{\text{opt}}, f_{\boldsymbol{\varepsilon},q}^{\text{opt}} \right)$ by the result of problem in step 3.2. (The best particle index, q, can change or not.)

3.5.2. Reset the survival index: $m \leftarrow 0$.

3.6. If $K_{\min} < k < K_{\max}$:

3.6.1. Update the elite group $\boldsymbol{\mathcal{B}}^k = \left\{ \left(\mathbf{x}_p^{\text{opt},k}, \mathbf{v}_p^{\text{opt},k} \right) \right\}_{p \in \overline{1,P}}$ and the crowd group $\boldsymbol{\mathcal{W}}^k = \left\{ \left(\mathbf{x}_p^{\text{tpo},k}, \mathbf{v}_p^{\text{tpo},k} \right) \right\}_{p \in \overline{1,P}}$.

3.6.2. Compute the swarm variance with the help of definition [2.62]:

$$\sigma_{\boldsymbol{\varepsilon}}^k = \frac{1}{P} \sum_{p=1}^{P} \left\| \mathbf{x}_p^k - \mathbf{x}_{\boldsymbol{\varepsilon},q}^{\text{opt}} \right\|^2 .$$

3.6.3. Compute the swarm relative variance with the help of definition [2.63]:

$$\rho_{\boldsymbol{\varepsilon}}^k = \frac{\sigma_{\boldsymbol{\varepsilon}}^k - \sigma_{\boldsymbol{\varepsilon}}^{k-1}}{\sigma_{\boldsymbol{\varepsilon}}^{k-1}} .$$

3.6.4. Perform ranking inside both the elite group $\boldsymbol{\mathcal{B}}^k$ and the current generation $\boldsymbol{\mathcal{E}}^k$ (in decreasing order of particles performance: the best one is the first one too).

3.6.5. If $\rho_{\boldsymbol{\mathcal{E}}}^{k} > \alpha$, the swarm is too scattered and has to be regrouped:

3.6.5.1. Initialize the temporary generation $\boldsymbol{\mathcal{E}}^{\alpha}$ by the P_e first particles from generation $\boldsymbol{\mathcal{E}}^{k}$. (Do not change the generation $\boldsymbol{\mathcal{E}}^{k}$ though.) Since $\boldsymbol{\mathcal{E}}^{\alpha}$ finally has to include P particles, the vacant position are to be taken by applying crossover between particles.

3.6.5.2. While $\boldsymbol{\mathcal{E}}^{\alpha}$ still is offering vacant positions, do:

a. Use a U-PSRG to select a particle from the current generation $\boldsymbol{\mathcal{E}}^{k}$, say $(\mathbf{x}_1, \mathbf{v}_1)$, as the first parent.

b. Use a U-PSRG to select a particle from the current elite group $\boldsymbol{\mathcal{E}}^{k}$, say $(\mathbf{x}_2, \mathbf{v}_2)$, as the second parent.

If necessary, repeat the selection until the two parents are different to each other.

c. Perform crossover between the two parents above, by using the equations [2.66], [2.64] and [2.65]. To do so, compute first the mating probability:

$$\gamma = \frac{1}{2} - \frac{\langle \mathbf{v}_1, \mathbf{v}_2 \rangle}{2\|\mathbf{v}_1\|\|\mathbf{v}_2\|},$$

then the offspring positions and speeds:

$$\begin{cases} \mathbf{y}_1 = \gamma \cdot \mathbf{x}_1 + (1-\gamma) \cdot \mathbf{x}_2 \\ \mathbf{y}_2 = \gamma \cdot \mathbf{x}_2 + (1-\gamma) \cdot \mathbf{x}_1 \end{cases}; \quad \begin{cases} \mathbf{w}_1 = \gamma \cdot \mathbf{v}_1 + (1-\gamma) \cdot \mathbf{v}_2 \\ \mathbf{w}_2 = \gamma \cdot \mathbf{v}_2 + (1-\gamma) \cdot \mathbf{v}_1 \end{cases}.$$

d. Add to the temporary generation $\boldsymbol{\mathcal{E}}^{\alpha}$ the best particle among the two parents and the two offspring as previously generated. (Do not change $\boldsymbol{\mathcal{E}}^{\alpha}$ if that best particle already belongs to it.)

e. For the added particle:

- set the history index (the "k_p") to the current iteration index, k;
- initialize $\mathbf{x}_*^{\text{opt}} = \mathbf{x}_*^{\text{tpo}}$ with the particle position and $f_*^{\text{opt}} = f_*^{\text{tpo}}$ with its performance.

3.6.5.3. Replace the current generation $\boldsymbol{\mathcal{E}}^{k}$ by the complete temporary generation, $\boldsymbol{\mathcal{E}}^{\alpha}$.

3.6.6. If $\rho_{\boldsymbol{\mathcal{E}}}^{k} < -\alpha$, the swarm is too concentrated and has to be dispersed:

3.6.6.1. Initialize the temporary generation $\boldsymbol{\mathcal{E}}^{\alpha}$ by the P_e first particles from generation $\boldsymbol{\mathcal{E}}^{k}$. (Do not change the generation $\boldsymbol{\mathcal{E}}^{k}$ though.) Since $\boldsymbol{\mathcal{E}}^{\alpha}$ finally has to include P particles, the vacant position are to be taken by applying crossover between particles.

3.6.6.2. While at least one of the P_c positions in $\boldsymbol{\mathcal{E}}^{\alpha}$ still is vacant, do:

a. Use a U-PSRG to select a particle from the current generation $\boldsymbol{\mathcal{E}}^{k}$, say $(\mathbf{x}_1, \mathbf{v}_1)$, as the first parent.

b. Use a U-PSRG to select a particle from the current crowd group $\boldsymbol{\mathcal{W}}^{k}$, say $(\mathbf{x}_2, \mathbf{v}_2)$, as the second parent. If necessary, repeat the selection until the two parents are different to each other.

c. Perform crossover between the two parents above, according to step 3.6.5.2.c.

d. Add to the temporary generation $\boldsymbol{\mathcal{E}}^{\alpha}$ the best particle among the two parents and the two offspring as previously generated. (Do not change $\boldsymbol{\mathcal{E}}^{\alpha}$ if that best particle already belongs to it.)

e. For the added particle:

- set the history index (the "k_p") to the current iteration index, k;
- initialize $\mathbf{x}_{*}^{\text{opt}} = \mathbf{x}_{*}^{\text{tpo}}$ with the particle position and $f_{*}^{\text{opt}} = f_{*}^{\text{tpo}}$ with its performance.

3.6.6.3. While $\boldsymbol{\mathcal{E}}^{\alpha}$ still is offering vacant positions, do:

a. Use a U-PRSG to select at random a position form the search space $\boldsymbol{\mathcal{S}}$. The position has to be different from all the positions that the swarm particles may have recorded within their histories. (If necessary, repeat this operation until this requirement is met.)

b. Use a U-PRSG to select a correspondent speed vector at random.

c. Add the couple position-speed to the temporary generation $\boldsymbol{\mathcal{E}}^{\alpha}$.

d. For the added particle:
- set the history index (the "k_p") to the current iteration index, k;
- initialize $\mathbf{x}_*^{opt} = \mathbf{x}_*^{tpo}$ with the particle position and $f_*^{opt} = f_*^{tpo}$ with its performance.

3.6.6.4. Replace the current generation $\mathcal{E}^k$ by the complete temporary generation, $\mathcal{E}^\alpha$.

4. Return:

➢ The best solution in the swarm: $\left(\mathbf{x}_{\mathcal{E},q}^{opt}, f_{\mathcal{E},q}^{opt}\right)$.

Obviously, the pseudo-code of Algorithm 2.9 above is not necessarily the most efficient one, but the goal was here to clearly describe all the procedure steps. If the implementation of Algorithm 2.9 (also referred to as *adaptive PSOA* (APSOA)) is professionally performed (suitably, on a parallel machine), then the search for the optimum can be taking quite short time (few tens of iterations), even in case of criteria with strong fractal nature. Moreover, in the most applications, there is a high probability to find an approximation of the global optimum (or even the global optimum itself). Nevertheless, the APSOA requires quite a large amount of memory to store and update every particle evolution history (see the step 3.2.18).

In general, the APSOA is one of the most efficient global metaheuristics, which can be adapted to a large panoply of granular optimization problems. The procedure does not require a special structure of the search space (like in case of ACA), nor a specific representation of the search space points (like in case of GA).

An application in which the APSOA has been integrated is described in the end of this section.

2.5.4. *Fireflies algorithm*

2.5.4.1. *Principle*

The fireflies are small flying insects belonging to Coleoptera order. They cyclically cast cold light signals by night, on mutual attraction purpose. In nature, the females can send light signals in order to attract males, which, after mating, are being captured and eaten. The genuine optimization algorithm of fireflies was introduced by Xin-She Yang in [YAN 08], after being inspired by the phenomenon of natural light attenuation depending on distance and by the mutual attraction between such insects. The algorithm works with sexless fireflies, though.

The fireflies are grouped into a swarm $\mathcal{L}$. Each of them takes a position in the search space $\mathcal{S}$ and can move to another position, depending on the ability to detect light signals the other fireflies are casting. In fact, each firefly is attracted by shinier fireflies. The attractiveness is proportional to the light intensity, which, in turn, decreases with the distance.

The fireflies are moving at random and their brilliance can be put into correspondence to the optimization criterion values. If the criterion has to be maximized, then the shiniest firefly location can be an optimal point. In case of minimization, the brilliance usually is inversely proportional (or opposite) to the criterion, so that the shinier firefly also can point to an optimal solution.

The Figure 2.18 constitutes a good illustration of fireflies swarm behavior as well. However, this time, it is possible that, in the end, the fireflies are grouped around several shiniest mates. Therefore, the fireflies based metaheuristic has a high degree of parallelism, as the swarm naturally keeps certain diversity from the beginning to the end of search. Thus, the global optimum of f criterion is more likely to be found, even in case of stochastic criteria, which is an important advantage. The main drawback is that the user cannot control very well the exploration-exploitation trade-off. Even more, the firefly metaheuristic is practically slightly unbalanced towards exploration almost all the time during the search.

2.5.4.2. *Dynamical model of fireflies behavior*

Consider that the swarm $\mathcal{L}$ includes P fireflies and let $k \in \mathbb{N}$ be the index of current iteration within the corresponding optimization algorithm. Then, according to the principle of perceived light attractiveness, the firefly $p \in \overline{1,P}$ is flying towards the shinier firefly $q \in \overline{1,P}$, its new position being evaluated as below:

$$\mathbf{x}_p^{k+1} = \mathbf{x}_p^k + \beta_{p,q}^k \left(\mathbf{x}_q^k - \mathbf{x}_p^k \right) + \alpha^k \operatorname{sign}\left(\rho^k - \frac{1}{2} \right) \xi^k \left(\phi_{\mathcal{S}} \right). \qquad [2.67]$$

The following new notations are employed in recursive equation [2.67] (beside the well known ones):

- $\beta_{p,q}^k$ is the intensity of light coming from the q firefly, as perceived by the p firefly;
- $\alpha^k \in [0,1]$ expresses the influence of the other fireflies in the swarm during the flight of current firefly to the shinier firefly; the initial direction of flight can be changes as result of other perceived lights on the way towards the shinier firefly; numerically, α^k is a (pseudo-)random number with uniform distribution;
- $\operatorname{sign}(a)$ is the sign of $a \in \mathbb{R}$;
- $\rho^k \in [0,1]$ is a number at random with uniform distribution; practically, $\operatorname{sign}\left(\rho^k - \frac{1}{2} \right)$ is generating a (pseudo-)random sign with uniform distribution;

- $\xi^k(\phi_{\mathcal{S}}) \in \mathbb{R}^{nx}$ is a vector parameter set at random (with a certain distribution, usually non uniform), which stands for the disturbing direction affecting the main direction of firefly, during its flight; of course, this perturbation appears as result of the other fireflies in the swarm, also casting light;
- $\phi_{\mathcal{S}}$ is an extremely important parameter for the algorithm convergence (although this phenomenon cannot soundly be proven); thus, $\phi_{\mathcal{S}}$ stands for the search space *diameter* or the *representation scale* of the optimization framework.

For more clarity, the terms of recursive equation [2.67] are explained next at length.

The expression of perceived light intensity, $\beta_{p,q}^k$, relies on the distance between two fireflies, simply denoted by $\mathrm{d}\left(\mathbf{x}_p^k, \mathbf{x}_q^k\right)$. Actually, it is about the Euclidean distance between the two positions the fireflies are located in:

$$\mathrm{d}\left(\mathbf{x}_p, \mathbf{x}_q\right) = \left\|\mathbf{x}_p - \mathbf{x}_q\right\|, \quad \forall \mathbf{x}_p, \mathbf{x}_q \in \mathcal{S}. \qquad [2.68]$$

(Of course, the distance depends on the search space topology and can be non Euclidean, if necessary.) The light intensity decreases when the distance increases. The biologists who observed the fireflies behavior, concluded that a possible model concerning the way they perceive light signals is described by the following equation:

$$\beta_{p,q} = I_0 \exp\left[-\gamma\, \mathrm{d}^2\left(\mathbf{x}_p, \mathbf{x}_q\right)\right] = I_0 \exp\left[-\gamma \left\|\mathbf{x}_p - \mathbf{x}_q\right\|^2\right], \quad \forall \mathbf{x}_p, \mathbf{x}_q \in \mathcal{S}. \qquad [2.69]$$

In equation [2.69], I_0 is the ideal relative light intensity, as perceived by the firefly if no distance would separate it from the other fireflies. Also, $\gamma > 0$ quantifies the attractiveness within the swarm, depending on the fireflies types and the environmental conditions (especially referring to the light absorption characteristic). Since I_0 is only relative, the unit value suits well as arbitrary choice ($I_0 = 1$). But the γ parameter is a key factor to manage the convergence speed of the metaheuristic. (It also can help to control the exploration-exploitation trade-off.) Although γ theoretically varies in a wide range, from null to infinity, its normal variation is bounded to the interval $\left[0, \phi_{\mathcal{S}}\right]$, which actually gives the search space representation scale.

The search space diameter is defined as below:

$$\phi_{\mathcal{S}} = \sup_{\mathbf{x},\mathbf{y} \in \mathcal{S}} \left\{\mathrm{d}\left(\mathbf{x}, \mathbf{y}\right)\right\}. \qquad [2.70]$$

Sometimes, computing the diameter following the definition [2.70] is far from easy, especially when the search space is highly irregular in shape. (Imagine, for example, a search space similar to a chestnut shell.) For this reason, in engineering, the diameter is determined quite coarsely, by considering the minimum volume hyper-sphere of $\mathbb{R}^{nx}$ that includes the search space $\mathcal{S}$. More important than the diameter are the representations scales inside the search space $\mathcal{S}$. For example, in case of rectangular shape, assume that $\mathcal{S}=[0,10]\times[0,100]\times[0,1000]$. Then the $0x$ axis is 10 times smaller than the $0y$ axis, which, on its turn, is 10 times smaller than the $0z$ axis. In this case, the fireflies positions have different sensitivities along the axes of search space. This feature has to be accounted, in order to avoid applying too many times the making viable strategy on new positions (which would destroy the search strategy). Therefore, if possible, instead of working with search space diameter, it is suitable to employ a set of scales, such as $\{\sigma_i>0\}_{i\in\overline{1,nx}}$, which shall be assigned to components of position $\mathbf{x}\in\mathcal{S}$. More specifically, the variation of x_1 is included into the interval, that of x_2 – into the interval $[0,\sigma_2]$, etc.

Usually the γ parameter is set as:

$$\gamma=\frac{1}{\sqrt{\phi_{\mathcal{S}}}}\quad\text{or}\quad\gamma=\frac{1}{\sqrt{\max\limits_{i\in\overline{1,nx}}\{\sigma_i\}}}. \qquad [2.71]$$

The last term of equation [2.67] (the third one) is the random engine of the search based on fireflies.

During the flight towards the shiniest detected firefly, q, some other fireflies of the swarm are detected. They can confuse the firefly p, which is thus tempted to change the flight direction, as being set in the beginning. The strength of this temptation is quantified by the factor $\alpha^k\in[0,1]$. If α^k is quite small (approaching the null value), then the firefly p rather is conservative and follows its initial direction towards the firefly q with very small influence from the swarm (as the second term of equation [2.67] dominates the position updating process). On the contrary, if α^k approaches the unit values, then the firefly p is not that determined to continue on its initial path, being tempted to adventure other paths, towards fireflies met on the fly (as the weight of second term in equation [2.67] can be compensated and even over-passed by the weight of the thirds term). Thus, on its way, the firefly is faced to at least two temptations.

The sign preceding the vector ξ^k expresses in fact a natural firefly characteristic: during the flight, it can suddenly change of direction towards another recently discovered and more attractive firefly.

Concerning the disturbing direction $\boldsymbol{\xi}^k$, it normally should vary in a range which is well adapted to the search space scales. Thus:

$$\boldsymbol{\xi}^k \in \left[0, \boldsymbol{\phi}_{\boldsymbol{s}}\right]^{nx} \quad \text{or} \quad \boldsymbol{\xi}^k \in \prod_{i=1}^{nx} \left[0, \sigma_i\right]. \tag{2.72}$$

Its components are selected according to some probability distribution $\boldsymbol{\wp}$, according to the dynamical behavior of the entire fireflies swarm. More specifically, the component ξ_i $(i \in \overline{1, nx})$ is generated in the interval $\left[0, \boldsymbol{\phi}_{\boldsymbol{s}}\right]$ or, if possible, $\left[0, \sigma_i\right]$, with the probability determined by distribution $\boldsymbol{\wp}$. For example, the distribution can be set as normal (Gaussian), according to the Central Limit Theorem. It follows that a N-PRSG (based on the BGA) can be employed to generate the component ξ_i. (In Appendix B, there is an example that shows hot to adapt the normal probability distribution to the selection set.) In case of fireflies, the selection sets are either $\left[0, \boldsymbol{\phi}_{\boldsymbol{s}}\right]$ or $\left\{\left[0, \sigma_i\right]\right\}_{i \in \overline{1, nx}}$. Since the sets are contiguous, they have to be sampled with some finesse (which, in fact, gives the roulette resolution within the GBA.)

A probability distribution coming from Biology seems to better match the fireflies behavior, namely the Lévy's distribution, defined as below [YAN 10a]:

$$\boldsymbol{\wp}(u) = u^{-\lambda}, \quad \forall u \in \mathbb{R}, \tag{2.73}$$

where $\lambda \in [1,3]$ is a parameter depending on the fireflies types in the swarm. (A recommended value of this parameter is $\lambda = 1.5$ [YAN 10a].)

In applications, an equivalent expression to [2.67] is sometimes employed:

$$\mathbf{x}_p^{k+1} = \left(1 - \beta_{p,q}^k\right)\mathbf{x}_p^k + \beta_{p,q}^k \mathbf{x}_q^k + \alpha^k \operatorname{sign}\left(\rho^k - \frac{1}{2}\right)\boldsymbol{\xi}^k\left(\boldsymbol{\phi}_{\boldsymbol{s}}\right), \tag{2.74}$$

in order to emphasize the new *purely conservative* position of firefly p (i.e. without considering the disturbing direction). This conservative position lies on the segment between the current position of firefly p and the position of target firefly q. (If $I_0 = 1$, the definition [2.69] shows that the perceived light intensity, $\beta_{p,q}^k$, varies in the $(0,1]$ interval.)

Like in case of particle swarms, the new positions [2.67] or [2.74] have to be made viable, if necessary (by means of the same techniques).

The fireflies are stopped to fly when the number of iterations has touched its upper limit, namely $K_{\max}$. The solution to the optimization problem is then returned by *the best firefly* in the swarm, e.g. by the shiniest one. (Recall that the firefly brilliance closely depends on the optimization criterion f.)

2.5.4.3. *Standard fireflies algorithm*

The standard optimization procedure based upon fireflies behavior (also known as the *Fireflies Algorithm* (FA)) is described in Algorithm 2.10 (with some improvements, comparing to the original procedure introduced in [YAN 08]).

Algorithm 2.10. *Standard optimization procedure by using fireflies.*

1. Input data:

- Search space $\mathcal{S} \subseteq \mathbb{R}^{nx}$, seen as a geographical area where fireflies can move freely. If possible, indicate the representation scales $\{\sigma_i\}_{i\in\overline{1,nx}}$ or, at least, the diameter $\phi_{\mathcal{S}}$.
- Optimization criterion f (its definition) and type of optimum to search (minimum or maximum)
- Expression of the light intensity I, corresponding to the fireflies brilliance. Usually, $I \equiv f$ for maximization or $I \in \{-f, 1/f\}$ for minimization, although different definitions are possible.
- Configuring parameters:
  - the number of fireflies in the swarm, $P \in \mathbb{N}^*$ (usually, of few tens);
  - the attractiveness factor, $\gamma > 0$ (corresponding to the representation scale, as suggested in [2.71]; by default: $\gamma = 1$);
  - the probability density of disturbing directions, $\wp$ (by default: the Lévy's distribution [2.73], with $\lambda = 1.5$);
  - the minimum resolution of roulette within BGA, when employed to generate disturbing directions, $N \in \mathbb{N}^*$ (by default: $N = 1000$, i.e. the selection set of type $[a,b] \subset \mathbb{R}$ has to include at least 1000 uniformly distributed points);
  - the maximum number of iterations, $K_{\max} \in \mathbb{N}^*$.

2. Initialization

a. Distribute the P fireflies on the departure positions $\{\mathbf{x}_p^0\}_{p\in\overline{1,P}} \subset \mathcal{S}$. (If no preferred technique exists, use a U-PRSG to uniformly select the positions from the search space.)

b. Compute the light intensities of fireflies: $\{I(\mathbf{x}_p^0)\}_{p\in\overline{1,P}}$.

c. Detect the shinier firefly, i.e. initialize the optimal solution: $(\mathbf{x}_{\mathcal{L}}^{\text{opt}}, I(\mathbf{x}_{\mathcal{L}}^{\text{opt}}))$ (where $\mathcal{L}$ is the swarm of fireflies).

3. For $k \in \overline{0, K_{\max}-1}$ (where k is the iterations index):

3.1. For each firefly $p \in \overline{1,P}$:

3.1.1. Evaluate the disturbing direction and its intensity, as follows:

3.1.1.1. Use a U-PRSG of resolution at least equal to N, to generate the influence factor $\alpha \in [0,1]$.

3.1.1.2. Use a U-PRSG of resolution at least equal to N, to generate the sign parameter $\rho \in [0,1]$.

3.1.1.3. Use a P-PRSG of resolution at least equal to N, with the probability distribution ϕ_S, to generate the disturbing direction $\xi \in \left[0, \phi_S\right]^{nx}$ or $\xi \in \prod_{i=1}^{nx} [0, \sigma_i]$. (Adapt the probability distribution to the selection sets, before starting the BGA.)

3.1.1.4. Evaluate the virtual position the disturbing direction is pointing to:

$$\mathbf{x}_p^{k,k+1} = \alpha \cdot \operatorname{sign}\left(\rho - \frac{1}{2}\right) \cdot \xi .$$

3.1.2. For each possible target-firefly $q \in \overline{1,P}$ (with $q \neq p$):

3.1.2.1. If the target is casting a light beam with the same or higher intensity of the one of current firefly, i.e. if $I\left(\mathbf{x}_q^k\right) \geq I\left(\mathbf{x}_p^k\right)$, then:

a. Evaluate the light intensity that the current firefly is perceiving, by using the definition [2.69]:

$$\beta = I_0 \exp\left[-\gamma \left\| \mathbf{x}_p^k - \mathbf{x}_q^k \right\|^2\right] .$$

b. Evaluate the position the current firefly could touch when flying from position $\mathbf{x}_p^k$, by using the equation [2.67] or [2.74]. For example:

$$\mathbf{x}_p^{k,k+1} \leftarrow (1-\beta)\mathbf{x}_p^k + \beta \mathbf{x}_q^k + \mathbf{x}_p^{k,k+1} .$$

3.1.2.2. Otherwise, the current firefly can only be motivated to fly towards the virtual position, as determined by the disturbing direction:

$$\mathbf{x}_p^{k,k+1} \leftarrow \mathbf{x}_p^k + \mathbf{x}_p^{k,k+1} .$$

3.1.2.3. If necessary, make $\mathbf{x}_p^{k,k+1}$ viable, by a technique that accounts the search space topology. (If possible, gradually decrease each rebel component of $\mathbf{x}_p^{k,k+1}$.) The next firefly position $\mathbf{x}_p^{k+1}$ is thus obtained.

3.1.2.4. Update the intensity of light beam the firefly is casting in the new position: $I\left(\mathbf{x}_p^{k+1}\right)$.

3.1.2.5. If $I\left(\mathbf{x}_p^{k+1}\right) > I\left(\mathbf{x}_{\swarrow}^{\text{opt}}\right)$, then the optimal solution of fireflies swarm has to be updated:

$$\mathbf{x}_{\swarrow}^{\text{opt}} \leftarrow \mathbf{x}_p^{k+1}, \quad I\left(\mathbf{x}_{\swarrow}^{\text{opt}}\right) \leftarrow I\left(\mathbf{x}_p^{k+1}\right).$$

4. Return:

- The optimal solution pointed by the firefly that, on its journey, was the shiniest of all: $\left(\mathbf{x}_{\swarrow}^{\text{opt}}, f\left(\mathbf{x}_{\swarrow}^{\text{opt}}\right)\right)$ and $I\left(\mathbf{x}_{\swarrow}^{\text{opt}}\right)$ (if different from $f\left(\mathbf{x}_{\swarrow}^{\text{opt}}\right)$).

The procedure of Algorithm 2.10 is different from the genuine FA, as introduced in [YAN 08], although the same principle was applied. Thus, the shiniest firefly of the swarm is guided here by a realistic direction, which is computed by accounting the observations from Biology. This direction is not just uniformly generated at random in a preset range, but according to Lévy's distribution, coming from Biology. Anyways, the firefly cannot be blocked in some position, regardless its brilliance. (Thus, the firefly leaves even the shiniest state it may touch.) Therefore, the optimal solution is not necessarily pointed by the fireflies final positions. It is very likely that one of the fireflies has touched the optimal point somewhere on its path. Nevertheless, in the end, the fireflies are agglomerating some tight zones of the search space, where the best solutions were detected.

The step 3.1.1 of FA should be implemented with care, as the procedure can very easily turn into a plain Monte-Carlo search process. If the number of new positions that have to be made viable is larger than the number of new positions directly falling into the search space, then the fireflies swarm acts like a confused group, without strategy, by almost Brownian movements. This renders the FA inefficient (with slow convergence and a large number of optimization criterion evaluations).

An interesting analysis (in terms of efficiency) concerning the similarity between the FA and other metaheuristic strategies is developed in [YAN 10a]. For example, if $\gamma \to 0$, then each firefly is more and more perceiving the other fireflies as shinier that itself. In this case, the FA practically starts to become very similar to the PSOA. On the contrary, if $\gamma \to \infty$, then each firefly is flying in a more and more dense fog, with a weaker and weaker perception about the other fireflies in the swarm. Now, the FA rather is approaching the Monte-Carlo procedure.

According to some scientists, the FA is considered as more efficient than the PSOA and even than many other metaheuristics (e.g. based on bats or bees), especially in case of stochastic optimization criteria [LUK 09], [YAN 09], [YAN 10b], [AUN 11], [CHA 11].

Although the FA was only recently introduced, nowadays, more that 20 versions can be found into the literature. This metaheuristic already was integrated in a great number of applications from various fields. Some examples are given below:

- constraint optimization of task distribution [LUK 09] ;
- multi-modal optimization [YAN 09], [FAR 12] ;
- optimization of manufacturing processes [AUN 11] ;
- optimization in complex non linear problems [ABD 12] ;
- optimization in concentric antennas reconfiguration [CHA 12] ;
- optimization of charges distribution in an electrical grid [DEK 12] ;
- optimization in dynamical domains [NAS 12] ;
- optimization of an economic distribution valves system [YAN 12a].

2.5.5. *Bats algorithm*

2.5.5.1. *Principle*

The small bats essentially are eating insects, after detecting them through echolocation. They cast ultrasounds and, depending on the direction and intensity of the received return signals, they can locate the possible prey quite precisely. Moreover, bats have the amazing capacity to rapidly discriminate between preys and obstacles during their flight. By associating the bats behavior (in their quest for food) and the optimization criterion, an interesting metaheuristic is obtained, as described next.

According to biologists, the bats general behavior can be described as follows.

- In the beginning, the bat is flying blindly, over the search space, by casting batches of ultrasound impulses of certain *amplitude* (or *intensity*) and *rate* (or *density*).
- Between the batches, the bat perceives by echolocation the return signals (its own and of the other bats in the swarm, if any).
- Decoding of received signals is performed instantly. If such signals are weak in intensity and/or strong in rate, then, maybe, a prey has been detected and, after locating it, the bat starts flying in that direction. The explanation of this phenomenon is quite simple: the prey act like an absorbent for the ultrasounds, which sensibly decreases the intensity of return signals; simultaneously, the pray is receiving ultrasounds from various bats, and thus, in return, the signal rate increases.
- As the bat approaches the prey, the density of ultrasound impulses (i.e. the rate) gradually increases, while the intensity (i.e. the amplitude) gradually decreases.

- If the received signals are very weak in rate, the bat continues to fly blindly, without changing the amplitude and the rate of transmitted ultrasounds.
- The distance to the prey is estimated by varying the frequency of ultrasounds over quite a large bandwidth. Thus, by using the Doppler effect, the bat can locate the prey quite accurately, even during its flight. Consequently, the bat can control and adapt the flying speed as well.

The *Bats Algorithm* (BatA) (as inspired from the bats behavior) was introduced by Xin-She Yang (the author of FA as well) in [YAN 10b]. The modeling details leading to the BatA design are described in the sequel.

2.5.5.2. *Dynamical model of bats behavior*

Let $\mathscr{C}$ be a swarm of P bats seeking for food and denote by $k \in \mathbb{N}$ the iteration index within the corresponding optimization algorithm (as usual). Then the model of micro-waves propagation in an environment can be employed to express the flying speed of bat $p \in \overline{1,P}$ through the recursive equation below:

$$\mathbf{v}_p^{k+1} = \mathbf{v}_p^k + \left(\mathbf{x}_{\mathscr{C}}^{\text{opt},k} - \mathbf{x}_p^k\right)\varphi_p^k , \qquad [2.75]$$

where $\mathbf{x}_{\mathscr{C}}^{\text{opt},k}$ is the current optimal position within the swarm, $\mathbf{x}_p^k$ is the current position of bat and φ_p^k is the current frequency of ultrasound the bat is casting. On its turn, this frequency is updated through an equation that accounts for the biological limitations of bat, namely the minimum and maximum frequencies of transmitted ultrasound ($\varphi_p^{\min}$ and $\varphi_p^{\max}$, respectively). More specifically:

$$\varphi_p^k = \left(1-\beta_p^k\right)\varphi_p^{\min} + \beta_p^k\varphi_p^{\max} . \qquad [2.76]$$

Normally, the parameter $\beta \in [0,1]$ should be set according to the biological characteristics of bat. Unfortunately, such a model is difficult to design (as not enough information is coming from biologists yet). Therefore, this parameter is selected at random, which is not too far from reality, as the bat usually is changing the frequency all the time, mostly by instinct. Only when the prey is detected the frequency is varied systematically, but, here too, the variation law still remains unknown for the biologists.

In nature, the frequency bounds are no less than 25 kHz and no more than 150 kHz for the most types of bats. Very few of the bats are able to overpass the upper bound of 150 kHz, but some of them can even reach 500 kHz. (For the reader's information, the humans can only perceive sounds between few Hz up to 20 kHz, and, moreover, the most human beings are hearing nothing beyond 12 kHz up. The signals transmitted by the bats are thus inaudible for us.) Nevertheless, in order to diminish the necessity to make viable too many new bat positions during

the numerical procedure, the recommended frequency bounds are: $\varphi_p^{\min} = 0$ and $\varphi_p^{\max} = 100$. In fact, the φ_p^k parameter can be seen as a relative frequency varying in range 0–100% of the bat specific bandwidth. When analyzing the equation [2.76], a fact becomes obvious: like in case of fireflies, the frequency parameter has to be adapted to the representation scale of search space.

The speed expression [2.75] actually is modeling the essential of bats behavior: despite its current speed and direction, each bat changes of direction towards the location of possible prey, as pointed by the swarm of its companions (see the difference between the vector of optimal position and the vector of current position resulting in a new direction).

After the speed has been updated, the bat flies towards the new position that can be determined as follows:

$$\mathbf{x}_p^{k+1} = \mathbf{x}_p^k + \mathbf{v}_p^{k+1} \Delta T_p^{k+1}. \qquad [2.77]$$

In equation [2.77], like in case of PSOA, the flight duration (delay) ΔT_p^{k+1} is (pseudo-)randomly set in some interval $(0, T]$, with $T > 0$ a priori known. Obviously, here too, the T parameter and the successive delays ΔT_p^{k+1} can be used to make viable the new positions [2.77], if necessary, by the same techniques as in PSOA. (Recall that perhaps the best technique is to gradually decrease the rebel components of term $\mathbf{v}_p^{k+1} \Delta T_p^{k+1}$, if possible.)

The initialization of recursive equations [2.75]–[2.77] is randomly set (if possible, by a set of uniformly distributed departure locations on the search space). With the initial position $\mathbf{x}_p^0$ and speed $\mathbf{v}_p^0$, the next speed $\mathbf{v}_p^1$ is firstly computed (by means of equations [2.76] and [2.75]). Then, the next position $\mathbf{x}_p^1$ is evaluated (by means of equation [2.77]). This allows computing the speed $\mathbf{v}_p^2$ the bat will fly with, in order to reach the next position $\mathbf{x}_p^2$ and so on.

The updating equation [2.77] can only be employed if the bat has already detected a possible prey. If the bath is still flying blindly, in the quest for food, then its positions change differently. This time, a technique known as *random walk* is employed, as follows:

$$\mathbf{x}_p^{k+1} = \mathbf{x}_p^k + \xi_p^k \left\langle A \right\rangle_p^k. \qquad [2.78]$$

In definition [2.78], $\xi_p^k \in [-1, +1]$ is a (pseudo-)randomly selected number expressing the perception capacity of bat p, concerning the ultrasounds batches the

other bats of swarm are transmitting in the search space. Also, $\langle A \rangle_p^k$ is the average of transmitted ultrasound amplitudes. More specifically:

$$\langle A \rangle_p^k = \frac{1}{P-1} \sum_{\substack{q=1 \\ q \neq p}}^{P} A_q^k \,, \qquad [2.79]$$

where A_q^k is the amplitude of ultrasound coming from the bat $q \in \overline{1,P}$, different from bat p. The product $\xi_p^k \langle A \rangle_p^k$ is a scalar to add to each component of current position vector $\mathbf{x}_p^k$ of bat p. The effect is the displacement with some offset of this position, but, generally, not too far away. To make viable this position (if necessary), it suffices to gradually decrease the perception capacity ξ_p^k, in the sense that the next perception capacity always is generated in the interval $\left[-\left|\xi_p^k\right|, +\left|\xi_p^k\right|\right]$, as defined by the current perception capacity. Concerning the bat speed, it is recommended to preserve the current speed for the new position as well.

If the prey is detected, the bat has to decrease the ultrasounds amplitude (A_p^k) and, at the same time, to increase the batch rate (r_p^k). According to biologists, the two phenomena are modeled by the following equations:

$$A_p^{k+1} = \alpha \cdot A_p^k \quad \text{and} \quad r_p^k = r_p^0 \left[1 - \exp\left(-\gamma k\right)\right], \qquad [2.80]$$

where: $\alpha \in (0,1)$ is a constant expressing the ultrasounds relative attenuation, $r_p^0 \in (0,1)$ is the initial rate of sent ultrasounds and $\gamma > 0$ is another constant that quantifies the relative speed of ultrasounds rate increasing. Normally, the α and γ constants depend on biological characteristics of bats swarm. Since it is quite difficult to set such constants, they can be set empirically, after several runs of the BatA. For example, in [YAN 10b], the recommended values are $\alpha = \gamma = 0.9$ (although the tests were performed on smooth criteria without stochastic noises or fractal ruptures).

The recursive equations [2.80] need initializations. For the A_p^k amplitudes, one considers that they vary in a range from $A^{\max} \in [1,100]$ to $A^{\min} \in [0,1]$. These bounds have no effects on the absolute positions and speeds of bats. Therefore, they can easily be translated to the interval $\left[A^{\min} = 0, A^{\max} = 1\right]$. Obviously, $A^{\min} = 0$ describes an ideal situation in which the bat already has found the perfect prey in a position of the search space, being thus unmotivated to fly away. Anyways, the starting amplitudes, A_p^0, have to be chosen in a vicinity of $A^{\max}$, for each bat

$p \in \overline{1,P}$. Regarding the initial rates r_p^0, it is suitable to set them to quite small values, nearly null, in order to allow sufficient number of effective rate increasing. If r_p^0 is too big, very quickly, there will be no numerical difference between successive rates , as the computers are working with finite precision.

2.5.5.3. *Standard bats algorithm*

The standard optimization procedure founded on the bats behavior is summarized within the Algorithm 2.11.

Algorithm 2.11. *Standard optimization procedure by using bats.*

1. Input data:

- Search space $\mathcal{S} \subseteq \mathbb{R}^{nx}$, seen as a geographical area where bats can fly freely. If possible, indicate the representation scales along the search space axes.
- Optimization criterion f (its definition) and type of optimum to search (minimum or maximum)
- Configuring parameters:
  - the number of bats in the swarm, $P \in \mathbb{N}^*$ (usually, of few tens);
  - the frequency band of sent ultrasounds, $\left[\varphi_p^{\min}, \varphi_p^{\max}\right]$; it is recommended to express the bounds according to the representation scales of search space (by default, the relative bounds are: $\varphi_p^{\min} = 0$ and $\varphi_p^{\max} = 100$);
  - the maximum delay of bat flights, $T > 0$ (to be adapted to the representation scales of search space, if possible);
  - the ultrasounds attenuation relative speed, $\alpha \in (0,1)$ (by default: $\alpha = 0.9$);
  - the ultrasounds rate increasing relative speed, $\gamma > 0$ (by default: $\gamma = 0.9$);
  - the upper bound of ultrasounds amplitude, $A^{\max} > 0$ (by default: $A^{\max} = 100$);
  - the minimum resolution of roulette within BGA, when employed to generate disturbing directions, $N \in \mathbb{N}^*$ (by default: $N = 1000$, i.e. the selection set of type $[a,b] \subset \mathbb{R}$ has to include at least 1000 uniformly distributed points);
  - the maximum number of iterations, $K_{\max} \in \mathbb{N}^*$.

2. Initialization

a. Distribute the P bats on the departure positions $\left\{\mathbf{x}_p^0\right\}_{p\in\overline{1,P}} \subset \mathcal{S}$. (If no preferred technique exists, use a U-PRSG to uniformly select the positions from the search space.)

b. Use a U-PRSG of resolution at least equal to N, to set the initial speed of bats on the departure positions $\left\{\mathbf{v}_p^0\right\}_{p\in\overline{1,P}}$. (In order to make the correct choice, one has to account for the search space bounds (mainly determined by its diameter) and also for the maximum flights delay, T.)

c. Use a U-PRSG of resolution at least equal to N, to initialize the ultrasound amplitudes: $\left\{A_p^0\right\}_{p\in\overline{1,P}} \subset \left[0.95A^{\max}, A^{\max}\right]$.

d. Use a U-PRSG of resolution at least equal to N, to initialize the ultrasound rates, $\left\{r_p^0\right\}_{p\in\overline{1,P}}$ in the interval $\left[10^{-7}, 10^{-1}\right]$.

e. Estimate the bats performance: $\left\{f\left(\mathbf{x}_p^0\right)\right\}_{p\in\overline{1,P}}$.

f. Determine the closest bat to its prey, i.e. initialize the optimal solution: $\left(\mathbf{x}_{\mathcal{C}}^{\text{opt}}, f\left(\mathbf{x}_{\mathcal{C}}^{\text{opt}}\right)\right)$ (where $\mathcal{C}$ is the swarm of bats).

g. Send the batches of ultrasounds, for a first investigation of search space $\mathcal{S}$. (This casting is only virtually performed, as no numerical effects exist on the procedure.)

3. For $k \in \overline{0, K_{\max}-1}$ (where k is the iterations index):

3.1. For each bat $p \in \overline{1,P}$:

3.1.1. Use a U-PRSG of resolution at least equal to N, to select the rate of received ultrasounds: $r^{\text{per}} \in [0,1]$.

3.1.2. If the rate above is superior to the transmitted rate, i.e. if $r^{\text{per}} > r_p^k$, then the bat has to fly towards the optimal position (which presumably is hosting the prey). In order to do so:

3.1.2.1. Use a U-PRSG of resolution at least equal to N, to select the frequency factor, $\beta \in [0,1]$.

3.1.2.2. Estimate the frequency of transmitted ultrasound, with the help of [2.76]:

$$\varphi = (1-\beta)\varphi_p^{\min} + \beta\varphi_p^{\max}.$$

3.1.2.3. Evaluate the slight speed towards the optimal solution of the entire swarm, with the help of equation [2.75]:

$$\mathbf{v}_p^{k+1} = \mathbf{v}_p^k + \left(\mathbf{x}_{\mathcal{C}}^{\text{opt}} - \mathbf{x}_p^k\right)\cdot\varphi.$$

3.1.2.4. Use a U-PRSG of resolution at least equal to N, to select the flight delay, $\Delta T \in [0,T]$.

3.1.2.5. Estimate the possible next position, with the help of recursive equation [2.77]:

$$\mathbf{x}_p^{k,k+1} = \mathbf{x}_p^k + \mathbf{v}_p^{k+1}\Delta T \ .$$

3.1.2.6. If necessary, make $\mathbf{x}_p^{k,k+1}$ viable, by a technique that accounts the search space topology. (If possible, gradually decrease each rebel component of $\mathbf{x}_p^{k,k+1}$.) The next firefly position $\mathbf{x}_p^{k+1}$ is thus obtained.

3.1.3. Otherwise, the received ultrasounds rate is at most equal to the send ultrasounds rate and the bat has to blindly fly towards a pseudo-random position. To do so:

3.1.3.1. Use a U-PRSG of resolution at least equal to N, to select the current perception capacity of bat, $\xi \in [-1,1]$.

3.1.3.2. Compute the average amplitude of sent ultrasound by the other bats in the swarm, according to definition [2.79]:

$$\langle A \rangle = \frac{1}{P-1}\sum_{\substack{q=1 \\ q\neq p}}^{P} A_q^k \ .$$

3.1.3.3. Evaluate the possible next position of bat, with the help of equation [2.78]:

$$\mathbf{x}_p^{k,k+1} = \mathbf{x}_p^k + \xi\langle A \rangle \ .$$

3.1.3.4. Preserve the flight speed: $\mathbf{v}_p^{k+1} = \mathbf{v}_p^k$.

3.1.3.5. If necessary, make $\mathbf{x}_p^{k,k+1}$ viable by gradually decreasing the bat perception capacity. More specifically, repeat:

$$\xi \leftarrow \text{U-PRSG}\left(\left(-|\xi|,+|\xi|\right)\right),$$

until $\mathbf{x}_p^k + \xi\langle A \rangle \in \boldsymbol{\mathcal{S}}$. The next position of bat results then:

$$\mathbf{x}_p^{k+1} = \mathbf{x}_p^k + \xi\langle A \rangle \in \boldsymbol{\mathcal{S}} \ .$$

3.1.4. Evaluate the bat performance on the new position: $f\left(\mathbf{x}_p^{k+1}\right)$.

3.1.5. Use a U-PRSG of resolution at least equal to N, to select the amplitude of ultrasounds the bat is perceiving: $A^{\text{per}} \in \left[0, A^{\max}\right]$.

3.1.6. If the perceived amplitude is inferior to the amplitude of sent ultrasounds (i.e. if $A^{\text{per}} < A_p^k$) and if the bat lies in a better position (i.e. if $f\left(\mathbf{x}_p^{k+1}\right)$ is better than $f\left(\mathbf{x}_{\ell}^{\text{opt}}\right)$), then the bat is approaching a possible prey (not an obstacle). In this case, the bat has to search the prey more intensely. To do so:

3.1.6.1. Decrease the amplitude of sent ultrasounds, with the help of first equation in [2.80]:

$$A_p^{k+1} = \alpha \cdot A_p^k .$$

3.1.6.2. Increase the rate of sent ultrasounds, with the help of second equation in [2.80]:

$$r_p^{k+1} = r_p^0 \left[1 - \exp\left(-\gamma(k+1)\right)\right].$$

3.1.7. Otherwise, either the bat is faced with an obstacle or it is far away from any prey. In this case, conserve the amplitude and the rate of sent ultrasounds ($A_p^{k+1} = A_p^k$ and $r_p^{k+1} = r_p^k$).

3.1.8. If the bat has arrived in a better position (i.e. if $f\left(\mathbf{x}_p^{k+1}\right)$ is better than $f\left(\mathbf{x}_{\ell}^{\text{opt}}\right)$), the update the optimal solution of the swarm:

$$\mathbf{x}_{\ell}^{\text{opt}} \leftarrow \mathbf{x}_p^{k+1}, \quad f\left(\mathbf{x}_{\ell}^{\text{opt}}\right) \leftarrow f\left(\mathbf{x}_p^{k+1}\right).$$

4. Return:

- The optimal solution of bats swarm: $\left(\mathbf{x}_{\ell}^{\text{opt}}, f\left(\mathbf{x}_{\ell}^{\text{opt}}\right)\right)$.

The Algorithm 2.11 is combining the adventurous and panurgian tendencies of bats. If some prey is detected, the bat continues its flight in that direction (see the equations [2.75]–[2.77]). Otherwise, the bat is inquiring about a good flight direction among its companions in the swarm (see the equations [2.78]–[2.79]).

The user can only have an indirect control over the exploration-exploitation trade-off, especially through the α et γ parameters. In fact, the relative speed of ultrasounds attenuation, α, plays a similar role to the temperature in simulated annealing procedure. The relative speed of ultrasounds rate increase, γ, allows the user to acquire a certain balance between the adventure spirit and the panurgian temptation the bats are coping with. Nevertheless, because the user has no direct control over the convergence parameters, very often, the BatA has to be rerun

several times, in order to find the "good" values of configuring parameters, by learning. This is the main drawback of BatA.

In spite of its youth, the BatA has already been integrated into a series of applications, such as:

- ergonomic optimization of working stations in large offices [KHA 11] ;
- energy flow modeling, optimization and fuzzy classification in a gas generator [LEM 11] ;
- global optimization in various engineering problems [YAN 12b] ;
- automatic optimal classification of data [MIS 12].

Apparently, according to the results reported so far, the BatA can efficiently work in conjunction with other optimization strategies or optimal classification techniques. However, sometimes, in some applications, the BatA can be less efficient than other metaheuristics described in this book.

2.5.6. *Bees algorithm*

2.5.6.1. *Principle*

The idea of *Bees Algorithm* (BeeA) was introduced by D. Karaboga in [KAR 05] and subsequently developed by V. Teresko and A. Loengarov in [TER 05] or by D.T. Pham in [PHA 05] (together with other authors), [PHA 09] and [PHA 13] (together with M. Castellani). The BeeA is inspired by honey bee colonies in their quest for food [TER 05].

Unlike the previous insect swarms, the bee colony is not only characterized by a certain dynamics, but also by a cooperative behavior, based on some hierarchical social structure. For the heuristic optimization, it is interesting to model the part of the bee "society" who is in charge with the food detection and harvest collecting. This population actually comprises the *workers* of the bees colony and actually includes the great majority of it. Basically, there are two types of worker bees: the *scouts* (which are responsible for the discovery of the food source) and the *foragers* (who perform harvest collecting and transportation to the hive).

The dynamical and cooperative behavior of the worker bees is as follows:

- The scouts look for the best food source (flower patches), by randomly moving in the area surrounding the hive. (Note, however, that a bee can deviate up to 14 km from the hive, still being able to return home.)
- They return to the hive with their harvested food.
- They are then directed to a special area in the hive known as the "*dance floor*". Once being arrived in this place, each scout begins to interpret the *waggle dance* (a sort of dance presenting the collected harvest), whose main goal is to demonstrate the richness and quality of the nectar and pollen from the visited flower patches. Note that the bees are not quite interested in a large amount of poor quality food or in a small amount of high quality food.

In such cases, sending a forager army (or group) to the source flower patches is considered inefficiently.
- Through the waggle dance, the scout bee progressively discharges its harvest. In the end, it can return back to the foraging or harvest transport expedition.
- During the waggle dance, the most charged scouts with quality food attract around them other workers (onlookers) who are recruited then as foragers.
- Each scout communicates to its foragers the information about direction and distance to the found flower patch.
- The foragers who report a larger harvest than the most part of the scouts become scouts, while the scout who is not able to find some profitable flower patches for the bee colony or who report a low quality/quantity of food becomes forager.

The potentially rich patches in terms of food quality are explored both globally (especially by the scouts) and locally (especially by the foragers). The bee colony approach is not only cooperative but also "autocatalytic", in the sense that the information about the most profitable flower patches gradually increases the recruitment of the foragers who participate at the harvest.

2.5.6.2. *Dynamical and cooperative model of bees behavior*

In order to design a model of bees behavior that leads to optimization, it is better to clarify the specific problem that best matches this type of metaheuristic. The manner in which the natural bees behave suggests that one could define certain sites on the search space where the artificial bees will perform local search for the optimum. This implies the necessity of working with well-defined topology over the search space $\mathcal{S}$. For example, the Euclidean topology is a natural one, in which the sites of interest are rectangular or circular vicinities of some central points.

An optimization problem that suits the bee metaheuristic is the following:

$$\begin{cases} \min\limits_{\mathbf{x}\in\mathbb{R}^{nx}} f(\mathbf{x}) \\ \text{s.t.}: \begin{cases} x_i \in \left[x_i^{\min}, x_i^{\max} \right], \ \forall i \in \overline{1,nx} \\ g_l(\mathbf{x}) \le 0, \ \forall l \in \overline{1,L} \\ h_r(\mathbf{x}) = 0, \ \forall r \in \overline{1,R} \end{cases} \end{cases}. \qquad [2.81]$$

In [2.81], the criterion f could have continuous or discrete variation and/or even stochastic nature. The envelope of search space $\mathcal{S}$ is defined by the hyper-parallelepiped of the first group of constraints. Its specific boundaries are defined by the other two groups of constraints (which could miss). It is not mandatory then the search space $\mathcal{S}$ is convex, but this property, if verified, can help to reduce the computational burden. in case of maximization problems, the formulation is similar.

Starting from the f criterion and an artificial bee located in position $\mathbf{x} \in \mathcal{S} \subset \mathbb{R}^{nx}$, one can define the bee fitness as follows:

$$\mathcal{F}(\mathbf{x}) = \begin{cases} \dfrac{1}{1+f(\mathbf{x})} & \text{, si } f(\mathbf{x}) \geq 0 \\ 1-f(\mathbf{x}) & \text{, si } f(\mathbf{x}) < 0 \end{cases}. \qquad [2.82]$$

Thus, minimization of criterion f involves maximization of fitness $\mathcal{F}$ over the P bees in the colony (denoted by $\mathcal{B}$). In this context, the fitness value is associated to the quantity and quality of food that was deposited to the hive by any bee.

The following model takes into account the characteristics of the real bee behavior.

According to the fitness, the bees can be classified into 3 categories:

- *scouts*, that perform the exploration of the search space, in order to find the central points of exploited sites (with large values of fitness); they are in number of P_s;
- *elite foragers*, which produce the best fitness values on the working sites; their number is $P_{ef} = N_{ef} \cdot M_{es}$, where M_{es} is the number of the elite sites to be exploited and N_{ef} is the number of elite foragers on each elite site;
- *remaining foragers*, which produce fairly large values of the fitness on different exploited sites than the elite ones; they are in number of $P_{rf} = N_{rf} \cdot (M_{bs} - M_{es})$, where M_{bs} is the total number of the best sites to be exploited in the search space and N_{rf} is the number of the remaining foragers on each remaining site.

Following this classification, the number of the bees in the colony is then:

$$P = P_s + P_{ef} + P_{rf} = P_s + N_{ef} M_{es} + N_{rf} (M_{bs} - M_{es}). \qquad [2.83]$$

Moreover, the following constraints have to be verified:

$$M_{es} \leq M_{bs} < P_s, \qquad [2.84]$$

since the best sites are indicated by the scouts and the elite sites are selected from the best sites. The last (strict) inequality is useful to refresh the scout group with the best foragers. (Thus, $P_s - M_{bs} > 0$ scouts (the weakest) are replaced by the foragers with better fitness.) This inequality is necessary to avoid the bee colony to be trapped into the vicinity of a local optimum and, thus, to preserve the colony diversity.

Initially, the group of P_s scouts usually is quite uniformly distributed over the search space, in order to perform a first exploration. The initial scouts can also be located according to a certain distribution, if any preliminary information on the search space zones potentially including an optimal point is provided.

Computing the fitness values of the scouts actually means organizing and watching the waggle dance. The scouts are then ranked according to their fitness, by descending order. The best scouts indicate the M_{bs} sites that are worthwhile to be exploited. As of the other scouts, either they are degraded and thus become solitary foragers or they join back the scouts group.

The elite scouts, composed of the bees that produce the first M_{es} best fitness values, recruit N_{ef} foragers each, for the food transportation. Therefore, $N_{ef} M_{es}$ bees are used to exploit the elite sites of the search space.

The other $M_{bs} - M_{es}$ scouts recruit N_{bf} foragers each one, in order to harvest the discovered flower patches. Thus, the remaining sites are exploited by $N_{bf}\left(M_{bs} - M_{es}\right)$ bees in the colony. The $P_s - M_{bs}$ degraded scouts are randomly sent to the search space, as solitary foragers. It is desirable to send them to the areas that have not been explored yet and not to the best sites. They could even return to the sites they already have visited, but in other positions.

The foragers fly to the sites that are indicated by the scouts who have recruited them. Such a site is centered to the specific point the scout was informing about, which virtually is associated to a flower. After the foragers departure, the scouts wait for the harvest. All returning forager that has superior fitness value to that of its waiting scout, will replace that scout. Thus, the scout becomes forager. The best scout could be overthrown or not. If it keeps its position during more than M_s iterations, then the search can be stopped and the optimal solution can be returned by this scout. Similarly, the search stops if the number of iterations becomes too large, for example above $K_{\max}$. In this case, the best scout indicates the optimal solution.

Any new scout communicates the position of a different flower around which the site should be centered. A site is defined as follows. Consider that the scout points to the flower that is located in position $\mathbf{x}_e \in \mathcal{S}$. Then, a hyper-cube can be built around this flower, with quite a large dimension, for example $2A > 0$ (a priori set). More specifically, the site is defined by the following equations:

$$x_{e,i}^{\min} = \max\left\{x_{e,i} - A, x_i^{\min}\right\} \le x_i \le \min\left\{x_{e,i} + A, x_i^{\max}\right\} = x_{e,i}^{\max}, \quad \forall i \in \overline{1, nx}. \qquad [2.85]$$

(One takes into account the limits of the search space, or at least of its envelope.) Denote by $\mathcal{H}_A\left(\mathbf{x}_e\right)$ the site that corresponds to the flower in position $\mathbf{x}_e$.

Once they have arrived on their site, the foragers begin to randomly exploit the flowers, being distributed in various points of the hyper-cube. The position of any forager, say $\mathbf{y} \in \mathcal{H}_A(\mathbf{x}_e)$, can be chosen as follows:

$$y_i = \gamma_i x_{e,i}^{\max} + (1-\gamma_i) x_{e,i}^{\min}, \quad \forall i \in \overline{1,nx}, \qquad [2.86]$$

where $\gamma_i \in [0,1]$ is randomly chosen, with uniform distribution. One still wants that:

$$\gamma_i \neq \frac{x_{e,i}^{\max} - x_{e,i}}{x_{e,i}^{\max} - x_{e,i}^{\min}}, \quad \forall i \in \overline{1,nx}, \qquad [2.87]$$

in order to avoid the selection of central point $\mathbf{x}_e$. (Certain random values γ_i could be identical to the right term of [2.87], but not all of them.) The forager position [2.86] shall be made viable (if necessary), by an adapted technique to the search space scales and nature. (For example, if $\mathcal{S}$ is discrete, then the position [2.86] will be replaced with the nearest integer in $\mathcal{H}_a(\mathbf{x}_e)$.) Each point corresponds to a certain fitness, which constitutes the performance of the associated forager.

If no forager is capable to exceed the performance of its scout in an exploited site, then the site should be compressed, while keeping the number of foragers (N_{ef} or N_{bf}). Thus, the exploitation of the site is *intensified*, since the points (flowers) to visit become closer and closer. The site reduction is carried out in a simple and natural way: by reducing the size of the corresponding hyper-cube. More specifically, if $a^0 = A$ is the initial size parameter, then this parameter gradually decreases, as follows:

$$a^{n+1} = \alpha \cdot a^n, \quad \forall n \in \mathbb{N}, \qquad [2.88]$$

where the contraction factor $\alpha \in (0,1)$ is a priori specified (for example, $\alpha = 0.9$). However, the number of successive contractions of a site must not exceed a number of K_a iterations. If no visited flower is capable to improve the fitness of the scout, then the site should be abandoned, except for the best scout site, as long as it is not overthrown. For this reason, it seems natural to choose $K_a \geq M_s$.

In case of a site abandonment, for example the site $p \in \overline{1,M_{bs}}$, its foragers must be assigned to another site. In order to choose the central position of the new site, several strategies could be envisaged. The simplest strategy is to randomly choose this position, in the search domain. This strategy is appropriate in case of degraded and solitary scouts, but it can slow down the search of the best sites.

More effectively is to take into account other exploitation sites and to select a position in one of those sites. For each one of the $M_{bs}-1$ sites (other than the abandoned ones), one can randomly choose a forager, for example $\mathbf{y}_q$, with $q \in \overline{1, M_{bs}} \setminus \{p\}$. Then the new central position can be selected by using a P-PRSG based on BGA, with the following probability profile:

$$\wp\left(\mathbf{y}_q\right) = \frac{\mathcal{F}\left(\mathbf{y}_q\right)}{\sum_{j=1, j\neq p}^{M_{bs}} \mathcal{F}\left(\mathbf{y}_j\right)}, \quad \forall q \in \overline{1, M_{bs}} \setminus \{p\}. \qquad [2.89]$$

The components of this dynamic cooperative model are integrated in the flow diagram in Figure 2.20.

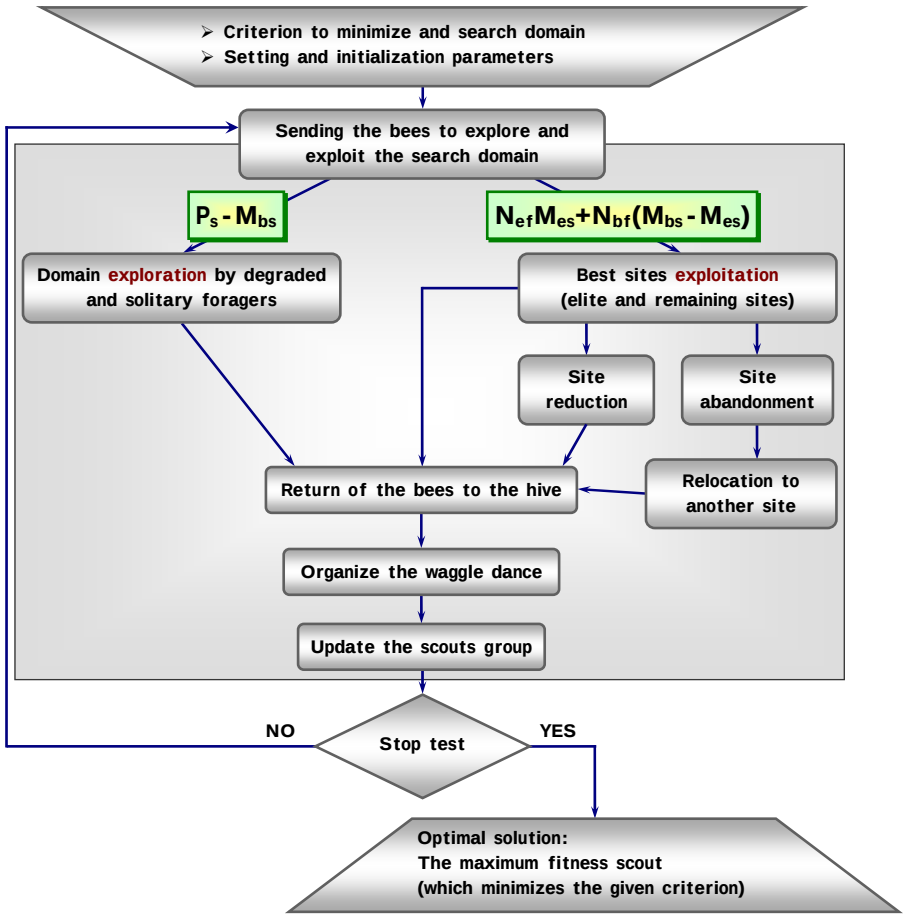


Figure 2.20. *Flow diagram of the bee colony algorithm.*

2.6.1.3. *Standard bee algorithm*

Following the diagram in Figure 2.20, the bee colony based procedure is summarized in Algorithm 2.12.

Algorithm 2.12. *Standard optimization procedure through bee colony.*

1. Input data:

- Optimization criterion to minimize, f (see the problem [2.81]).
- Constraint expressions, $\{g_l\}_{l\in\overline{1,L}}$ and $\{h_r\}_{r\in\overline{1,R}}$.
- Envelope of the search space, $\mathcal{S}\subseteq\mathbb{R}^{nx}$, seen as a hyper-parallelepiped, i.e. the limits : $\{x_i^{\min}\}_{i\in\overline{1,nx}}$ and $\{x_i^{\max}\}_{i\in\overline{1,nx}}$.
- The bees fitness expression, $\mathcal{F}$ (by default, definition [2.82] is employed).
- Configuring parameters:
  - the scouts number, $P_s\in\mathbb{N}^*$ (usually, between 100 and 1000);
  - the number of best sites to explore, $M_{bs}\in\mathbb{N}^*$, so that $M_{bs}<P_s$;
  - the number of the elite sites, $M_{es}\in\mathbb{N}^*$, so that $M_{es}\le M_{bs}$;
  - the number of foragers to be recruited for each elite site, $N_{ef}\in\mathbb{N}^*$;
  - the number of the foragers to be recruited for each one of the best remaining sites, $N_{rf}\in\mathbb{N}^*$;
  - the parameter referring to the initial size of the sites, $A\in\left(0,\frac{1}{2}\min_{i\in\overline{1,nx}}\{x_i^{\max}-x_i^{\min}\}\right]$ (by default, A is set to its maximum value);
  - the contraction coefficient of the sites, $\alpha\in(0,1)$ (by default: $\alpha=0.9$);
  - the minimum resolution of roulette within BGA, when employed to generate PRS, $N\in\mathbb{N}^*$ (by default: $N=1000$, i.e. the selection set of type $[a,b]\subset\mathbb{R}$ has to include at least 1000 uniformly distributed points);
  - the maximum number of iterations, $K_{\max}\in\mathbb{N}^*$;
  - the maximum number of iterations to try until the best solution of a site is found, $K_a\in\mathbb{N}^*$ (by default: $K_a=0.2\cdot K_{\max}$);
  - the survival factor of the best scout, $M_s\in\mathbb{N}^*$ (by default: $M_s=0.1\cdot K_{\max}$).

2. Initialization

a. Estimate the bees number in the colony, by using the equation [2.83]:

$$P = P_s + N_{ef} M_{es} + N_{rf} \left(M_{bs} - M_{es} \right).$$

b. Distribute the P_s scouts throughout the search domain $\left\{ \mathbf{x}_p^0 \right\}_{p \in \overline{1, P_s}} \subset \mathcal{S}$. (If there is no preferred technique, use a U-PRSG to perform a uniform distribution.)

c. Organize the first waggle dance, i.e., estimate the fitness of each bee and rank their group by fitness descending order:

$$\left\{ \mathbf{x}_1^0 \quad \mathbf{x}_2^0 \quad \cdots \quad \mathbf{x}_{P_s}^0 \right\}.$$
$$\mathcal{F}\left(\mathbf{x}_1^0 \right) \geq \mathcal{F}\left(\mathbf{x}_2^0 \right) \geq \cdots \geq \mathcal{F}\left(\mathbf{x}_{P_s}^0 \right)$$

d. Initialize the size parameters for the best sites to exploit: $\left\{ a_p = A \right\}_{p \in \overline{1, M_{bs}}}$.

e. Initialize the indices to measure for how many iterations the discovered sites by the scouts are not improved in terms of fitness:

$$na_p = 0, \quad \forall\, p \in \overline{1, P_s}.$$

f. Initialize the performance of bee colony (denoted by $\mathcal{B}$):

$$\mathbf{x}_{\mathcal{B}}^{\text{opt}} = \mathbf{x}_1^0, \quad f_{\mathcal{B}}^{\text{opt}} = f\left(\mathbf{x}_1^0 \right), \quad \mathcal{F}_{\mathcal{B}}^{\max} = \mathcal{F}\left(\mathbf{x}_1^0 \right).$$

g. Initialize the survival index of the best scout, $m = 0$.

3. For $k \in \overline{0, K_{\max} - 1}$ (where k is the iterations index) and while $m \leq M_s$, do:

3.1. Determine the central positions of the best sites, as communicated by the scouts: $\left\{ \mathbf{x}_p^{\text{opt},k} = \mathbf{x}_p^k \right\}_{p \in \overline{1, M_{bs}}}$.

3.2. For each best site, $p \in \overline{1, M_{bs}}$:

3.2.1. If $na_p > K_a$, the site must be abandoned, since no fitness improvement is obtained. In this case:

3.2.1.1. Use a U-PRSG for each of the best sites, other than the current one, in order to select a forager $\mathbf{y}_q$, with $q \in \overline{1, M_{bs}} \setminus \{p\}$.

3.2.1.2. Build the probability profile associated to the set of chosen foragers, by using the definition [2.89]:

$$\wp\left(\mathbf{y}_q\right)=\frac{\mathcal{F}\left(\mathbf{y}_q\right)}{\sum_{j=1,\,j\neq p}^{M_{bs}}\mathcal{F}\left(\mathbf{y}_j\right)},\quad \forall q\in\overline{1,M_{bs}}\setminus\{p\}.$$

3.2.1.3. Use a P-PRSG, based on BGA, with the previously estimated probability profile, in order to select the central point $\mathbf{x}_p^{\mathrm{opt},k}$ of the new site. Certainly, $\mathbf{x}_p^{\mathrm{opt},k}\in\left\{\mathbf{y}_q\right\}_{q\in\overline{1,M_{bs}}\setminus\{p\}}$. This point must be made viable, if necessary.

3.2.1.4. Reset the size parameter: $a_p=A$.

3.2.1.5. Reset the non improvement index: $na_p=0$.

3.2.2. Define the local search site by definitions [2.85]. In this aim, compute its limits along the axes:

$$\begin{cases}x_{p,i}^{\min}=\max\left\{x_{p,i}^{\mathrm{opt},k}-a_p,x_i^{\min}\right\}\\ x_{p,i}^{\max}=\min\left\{x_{p,i}^{\mathrm{opt},k}+a_p,x_i^{\max}\right\}\end{cases},\quad \forall i\in\overline{1,nx}.$$

3.2.3. If $p\le M_{es}$, an elite site was delimited. A U-PRSG of nx size is employed to select each one of the N_{ef} forager positions. To do so, one applies the definition [2.86], with the constraint [2.87]. For any forager in the site, its position is then:

$$y_i=\gamma_i x_{p,i}^{\max}+\left(1-\gamma_i\right)x_{p,i}^{\min},\quad \forall i\in\overline{1,nx},$$

where the set $\left\{\gamma_i\right\}_{i\in\overline{1,nx}}\subset[0,1]$ is returned by a U-PRSG of resolution at least equal to N. If $\mathbf{y}=\mathbf{x}_p^{\mathrm{opt},k}$, one of its compounds has to be modified, by repeating the selection until $\mathbf{y}\neq\mathbf{x}_p^{\mathrm{opt},k}$. Any above selected position has to be made viable, if necessary.

3.2.4. If $p>M_{es}$, a remaining site was delimited. A U-PRSG of nx size is employed to select each one of the N_{rf} forager positions. The positions are chosen as for the elite foragers (see the step 3.2.3).

3.2.5. Organize the waggle dance for the site members, after return to the hive. The aim of this preliminary dance is to determine the best forager on the site and its position, $\mathbf{x}_p^{k,k+1}$.

3.2.6. If the best forager fitness is at least equal to the fitness of scout that recruited the foragers for this site, i.e. if $\mathcal{F}\left(\mathbf{x}_p^{k,k+1}\right) \geq \mathcal{F}\left(\mathbf{x}_p^{\text{opt},k}\right)$, then:

3.2.6.1. Replace the corresponding scout by the best forager, which becomes thus the new scout. This involves: $\mathbf{x}_p^{k+1} = \mathbf{x}_p^{k,k+1}$.

3.2.6.2. Reset the size parameter: $a_p = A$.

3.2.6.3. Reset the non improvement index: $na_p = 0$.

3.2.7. Otherwise, the search has to be intensified throughout the site. To do so:

3.2.7.1. Preserve the corresponding scout: $\mathbf{x}_p^{k+1} = \mathbf{x}_p^k$.

3.2.7.2. Reduce the size parameter: $a_p \leftarrow \alpha \cdot a_p$.

3.2.7.3. Increase the non improvement index: $na_p \leftarrow na_p + 1$.

3.3. For each degraded scout, $p \in \overline{M_{bs}+1, P_e}$:

3.3.1. Use a U-PRSG of nx size and resolution at least equal to N, in order to select the new starting position of an isolated exploration. The definition [2.86] can be employed in this aim:

$$y_i = \gamma_i x_i^{\max} + \left(1-\gamma_i\right) x_i^{\min}, \quad \forall i \in \overline{1,nx},$$

where $\{\gamma_i\}_{i \in \overline{1,nx}} \subset [0,1]$ is the result of the U-PRSG. This time, the envelope boundaries of the search domain are used. Certainly, the new position $\mathbf{y}$ has to be made viable, if necessary.

3.3.2. When an isolated bee returns to the hive, it is added to the scouts group: $\mathbf{x}_p^{k+1} = \mathbf{y}$.

3.4. Organize the waggle dance for all scouts (including the isolated ones), i.e., rank their group by fitness decreasing order:

$$\left\{ \mathbf{x}_1^{k+1} \quad \mathbf{x}_2^{k+1} \quad \cdots \quad \mathbf{x}_{P_e}^{k+1} \right\}.$$
$$\mathcal{F}\left(\mathbf{x}_1^{k+1}\right) \geq \mathcal{F}\left(\mathbf{x}_2^{k+1}\right) \geq \cdots \geq \mathcal{F}\left(\mathbf{x}_{P_e}^{k+1}\right)$$

3.5. If the best scout was changed with respect to the previous iteration, i.e., if $\mathcal{F}_{\mathbf{x}}^{\max} \leq \mathcal{F}\left(\mathbf{x}_1^{k+1}\right)$ and $\mathbf{x}_{\mathbf{x}}^{\text{opt}} \neq \mathbf{x}_1^{k+1}$, then:

3.5.1. Update the bee colony performance:

$$\mathbf{x}_{\mathbf{x}}^{\text{opt}} = \mathbf{x}_1^{k+1}, \quad f_{\mathbf{x}}^{\text{opt}} = f\left(\mathbf{x}_1^{k+1}\right), \quad \mathcal{F}_{\mathbf{x}}^{\max} = \mathcal{F}\left(\mathbf{x}_1^{k+1}\right).$$

3.5.2. Reset the survival index: $m = 0$.

3.6. Otherwise, the best scout was not overthrown and its survival index has to be increased: $m \leftarrow m+1$.

4. Return:

- The optimal solution, as offered by the best scout in the bees colony: $\left(\mathbf{x}_{\mathcal{A}}^{\text{opt}}, f\left(\mathbf{x}_{\mathcal{A}}^{\text{opt}}\right)\right)$.
- If necessary, the maximal fitness: $\mathcal{F}_{\mathcal{A}}^{\max}$.

Like in case of ACA, more versions of Algorithm 2.12 have been published in the literature. The related numerical procedure is very stable (with few oscillations), but many resources (especially of memory) are necessary to complete the job. The user preserves a fairly direct control of exploration-exploitation trade-off by means of parameters M_{bs} and P_s. If M_{bs} is sensibly smaller than P_s, then the number of solitary bees is big and therefore the exploration is encouraged, to detriment of exploitation. On the contrary, if the user decides to work with few solitary bees, the exploitation is favored against the exploration. The colony diversity is also positively influenced by the site abandonment, provided that they are not too numerous. (In this case, the BeeA can oscillate, but this happens quite rarely.) In fact, the site relocation after the abandonment could be viewed as an evolution strategy by applying multiple crossovers between foragers, in order to select the new central position. In this case though, the offspring is identical to one of the parents (usually with the best fitness). Other evolutionary strategies can be considered to relocate the bees from a site to another.

Due to its advanced parallelism, the BeeA could efficiently be implemented on graphical parallel processor units, where its convergence speed would be unbeatable. Very rarely, a BeeA requires the help of other metaheuristics or a local optimization method, in order to succeed.

One can mention several applications where the BeeA was successfully applied:

- optimal object classification [PHA 10];
- manufacturing chain optimization [OZB 11], [XUW 12];
- optimal control [ALF 11], [FAH 12];
- optimization in biotechnology [BAH 08], [RUZ 13];
- multimodal optimization [SAY 09], [MAN 12].

2.6.2. *Multivariable prediction by PSO*

Many natural phenomena are observable by means of some released signals into the evolution environment. Some of such signals can be acquired for modeling purpose and especially to predict the phenomena evolution. Nowadays, one can find a broad class of powerful enough sensors for natural signals, such as: air and soil

humidity, temperature, solar radiation, the amount of rain, wind speed and direction, etc.

By knowing the dynamics of main ecological parameters, predictions with certain accuracy can be performed, in order to prevent the unwanted hazardous situations such as: floods, droughts, strong winds, etc. Beside the prevention of natural disasters, which could occur quite rarely, the predicted values can help to maintain or even improve a certain living condition in the ecological climate. For example, if a farm field or vineyard is watered at predicted instants, with an appropriate amount of water (predicted too), the crop can significantly increase. Similarly, the effectiveness of a wind turbine could increase if the wind speed can be predicted.

An accurate prediction can be obtained if two general requirements are met:

a. the acquired signals contain a reasonable noise level, which means the *signal-to-noise ratio* (SNR) has sufficiently large values;
b. the prediction models are accurate enough.

Data series acquired from a (natural or artificial) source during its evolution is referred to as *time series*. For the most natural phenomena, it is not enough to acquire a single time series, in order to construct a fairly accurate prediction model. In general, several time series that represent various cross-correlation parameters have to be acquired at the same time. In this case, one deals with *multivariable* time series.

A natural numeric model of a time series includes two additive components, reflecting two types of behaviors: a deterministic one and a stochastic one. The deterministic component expresses the trend (usually polynomial) and the seasonal nature of the phenomenon [STE 10b], [STE 13]. Other deterministic models can be considered as well, for example, by using wavelets [STE 09], [STE 10a]. Nevertheless, the deterministic component cannot take into account the cross-correlations between the various measured parameters. These correlations can be modeled only by using the stochastic component, for example of autoregressive type [SOD 89], [STE 05], [BOR 13].

An important issue in the numerical modeling of multivariable time series is the large number of potentially optimal prediction models. This is a natural effect of the fact that such models are parametric and both the values and the number of their coefficients are unknown. Fortunately, if the number of the coefficients is preset, different identification methods can be employed (generally based on the *Least Squares Method* (LSM), which is an optimization technique [BOR 13]), in order to estimate the unknown coefficients, with a fairly high accuracy. But for prediction, it is crucial to correctly choose the parameters number of the numerical model. The prediction can easily fail if this number is approximated in a coarse manner. Moreover, the prediction accuracy is usually very sensitive to the variation of this number. Finding the model that leads to maximum prediction accuracy constitutes a granular optimization problem that can be solved by means of a global metaheuristic.

In this section, an application for multivariable prediction of some greenhouse parameters is briefly described. The goal here is to improve the plants comfort, by using predictive control techniques. The application is detailed in [STE 10b] and [CUL 11]. In the sequel, the reader can see how an APSOA was used in order to solve the granular optimization problem associated to the estimation of a multivariable prediction model. (However, the problem of automatic control will not be referred here.)

In order to facilitate the understanding of this problem, consider first the case of a scalar time series. It corresponds to a sequence of N_y data (samples), $\{y[n] = y(nT_s)\}_{n\in\overline{1,N_y}}$, which were obtained by sampling with period T_s some environmental signal. Basically, this time series encodes two types of additive behaviors, deterministic (y_D) and stochastic (y_{AR}):

$$y \equiv y_D + y_{AR} . \qquad [2.90]$$

The latter is caused by various disturbances (or noises) that corrupt the measured data and cannot be avoided. If the stochastic component was null, then the time series would be perfectly predictable by means of its deterministic model y_D. Since this component cannot be removed, the prediction model attempts in fact to separate the two components, in a more or less accurate manner. As mentioned in the end of section 2.2, there is no tool that achieves a perfect separation of the useful signal from its corrupting noise. The prediction model accuracy however is significantly influenced by the quality of this separation.

The time series components are illustrated in the example of Figure 2.21.

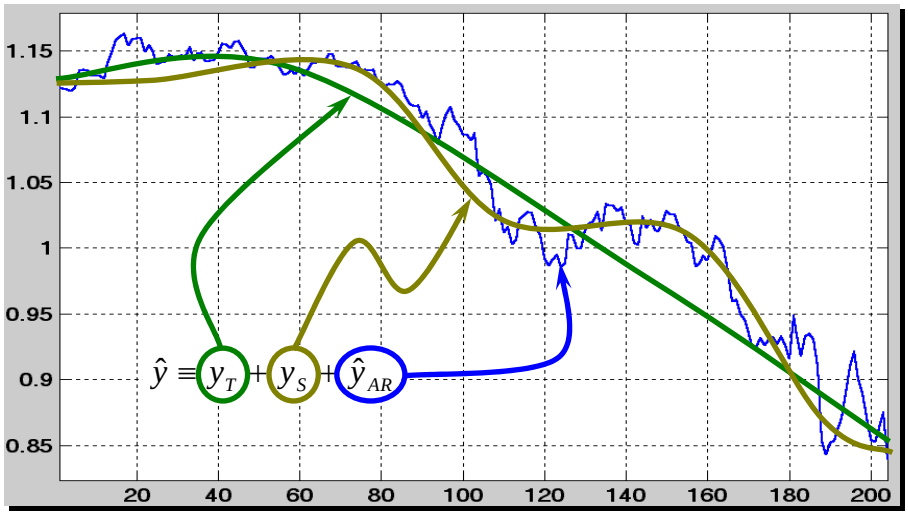


Figure 2.21. *Components of a time series.*

On its turn, the deterministic component is described by two models: a trend y_T and a seasonal variation y_S. The trend is of polynomial type:

$$y_T(t) = \alpha_0 + \alpha_1 t + \cdots + \alpha_{nt} t^{nt}, \quad \forall t \in \mathbb{R}, \qquad [2.91]$$

where both the coefficients $\{\alpha_i\}_{i\in\overline{1,nt}} \subset \mathbb{R}$ and their number, $nt+1 \in \mathbb{N}^*$, are unknown. The model [2.91] can be sampled with any period. The optimal coefficients can be determined, by using the LSM, if the degree nt is a priori known. This degree is often referred to as the *structural index* of model.

The seasonal model is determined by using spectral estimation methods (including Wittacker-Robinson and Schuster) [SOD 89], [STE 05]. This model does not add supplementary unknown parameters.

For multivariable time series, a data block $\mathbf{y}$ is defined, so that its components $\{y_j\}_{j\in\overline{1,ny}}$ correspond to the employed $ny \in \mathbb{N}^*$ measure channels. Thus, for each measure channel, a deterministic model has to be estimated.

The most interesting part consists of modeling the stochastic component. Two types of numerical models can be considered:

- of input-output type (from ARMAX class [SOD 89], [STE 05], [BOR 13]);
- with state space representation (based on Kalman-Bucy predictor [KAL 60], [KAL 61a], [KAL 61b]).

The generic model in the multivariable ARMAX class is the following:

$$\begin{cases} \mathbf{A}\left(q^{-1}\right)\mathbf{y}[n] = \mathbf{B}\left(q^{-1}\right)\mathbf{u}[n] + \mathbf{C}\left(q^{-1}\right)\mathbf{e}[n] \\ \mathrm{E}\left\{\mathbf{e}[n]\mathbf{e}^T[m]\right\} = \mathbf{\Lambda}\delta_0[n-m] \end{cases}, \quad \forall m,n \in \mathbb{N}, \qquad [2.92]$$

where: q^{-1} is one step delay operator ($\left(q^{-1}x\right)[n] = x[n-1]$, $\forall n \in \mathbb{Z}$);

$\mathbf{A} \in \mathbb{R}^{ny\times ny}\left(q^{-1}\right)$ is the first polynomial matrix in q^{-1}, with real coefficients, associated to the *autoregressive* part (AR) of the model;

$\mathbf{B} \in \mathbb{R}^{ny\times(ny-1)}\left(q^{-1}\right)$ is the second polynomial matrix in q^{-1}, with real coefficients, associated the *exogenous* part (X) of the model;

$\mathbf{C} \in \mathbb{R}^{ny\times ny}\left(q^{-1}\right)$ is the third polynomial matrix in q^{-1}, with real coefficients, associated to the *moving average* part (MA) of the model;

$\mathbf{u} \in \mathbb{R}^{ny-1}$ is the non measurable input vector of the model;

$\mathbf{y} \in \mathbb{R}^{ny}$ is the vector of all multivariable measurable outputs of the model;

$\mathbf{e} \in \mathbb{R}^{ny}$ is the vector of the stochastic noise (generally white, with Gaussian distribution), that corrupt the measured data, but that are not measurable as

well; notice that the number of perturbation channels equals to the number of measure channels (there are as many perturbation channels as the measure ones);

$\mathbf{\Lambda} \in \mathbb{R}^{ny \times ny}$ is a matrix that expresses the correlation degrees between the noise components (the *noise auto-covariance matrix*).

The degrees of the matrix polynomials (the structural indices, in fact) generally are different from each other and unknown. The coefficients of such polynomials are unknown too. Similarly, the auto-covariance matrix $\mathbf{\Lambda}$ is unknown. The inputs $\mathbf{u}$ of the model are in fact non-measurable noises, as, for each measure channel, the inputs are set from the noises of the other channels. What does one know about this model then? Only the measured outputs. However, if its structural indices are known, under certain assumptions, the model can be identified. During the identification, a simplified model is necessary, in order to estimate the noise values, based on the measured output values. This model adds more unknown parameters to the identification problem (coefficients and structural indices.) For example, if some ARMA model is selected (for which the matrix $\mathbf{B}$ is null), the auxiliary identification model will be of AR type (for which, in addition, the matrix $\mathbf{C}$ is unit).

From this model description it follows that the number of the structural indices to set is:

$$\underbrace{ny^2}_{\mathbf{A}(q^{-1})} + \underbrace{ny(ny-1)}_{\mathbf{B}(q^{-1})} + \underbrace{ny^2}_{\mathbf{C}(q^{-1})} + \underbrace{ny^2 + ny(ny-1)}_{\text{Auxiliary model}} = 5ny^2 - 2ny \,. \qquad [2.93]$$

If the ny structural indices of the trend are added, this number increases to $ny(5ny-1)$.

In general, the ARMAX models are identifiable for structural indices that vary in the range $\overline{0,\lfloor N_y/3 \rfloor}$ (where N_y is the number of measured data per channel). Sometimes, the ARMAX model cannot be identified with the structural indices in that range, but this is a very rare (but not impossible) situation. It follows that a good approximation of the number of available identification models is:

$$\mathcal{N} = \left(\left\lfloor \frac{N}{3} \right\rfloor + 1 \right)^{ny(5ny-1)} . \qquad [2.94]$$

For example, if N_y counts 1000 measured data per channel (which is a typical value in applications) and $ny = 3$ measure channels are available, then the number of prediction models to test is up to:

$$\mathcal{N} = 334^{42} , \qquad [2.95]$$

that is, more than 3.3×10^{44}. Since the identification and prediction of such a model can take up to 10 s, clearly, the offered prediction never completes on time, when exhaustive search is carried out in this huge set.

For the second class of models, of Kalman-Bucy type, a similar analysis can be developed. Although the total number of models to be tested is smaller, it still remains very large, which makes the exhaustive optimization impossible to use.

Returning to the prediction problem, the optimal models are selected by using a criterion known as the *prediction quality* (PQ). The PQ criterion can be defined as follows, for each measure channel [STE 10b]:

$$\mathrm{PQ}=\frac{100}{1+\dfrac{\sqrt{\sum_{k=1}^{K}\hat{\sigma}_k^2}}{\hat{\lambda}\sqrt{\mathrm{SNR}_{\mathrm{M}}}\sqrt{\mathrm{SNR}_{\mathrm{P}}}}}[\%]\,, \qquad [2.96]$$

where : $\{\hat{\sigma}_k\}_{k\in\overline{1,K}}$ are the estimated variances of the prediction errors, on a *prediction horizon* of $K\in\mathbb{N}^*$ instants (beyond the *measure horizon*);

$\hat{\lambda}$ is the estimated variance of the white noise for the corresponding measure channel;

$\mathrm{SNR}_{\mathrm{M}}$ is the estimated value of SNR on the measure horizon;

$\mathrm{SNR}_{\mathrm{P}}$ is the estimated value of SNR on the prediction horizon.

This criterion is expressed as percentage of acquired prediction performance and has to be maximized. It shows that a predictor is highly qualitative if its identification model performs a good separation between the useful component and the noise component of the measured data. However, its performance is limited by the noise level (the inverse of SNR) on the measure horizon. (Very noisy data practically are non predictable.)

In the case of multiple measure channels, the PQ values of each channel are grouped into a vector and thus one deals with a multi-criteria optimization problem. Sometimes, this problem is reduced to the scalar case, by considering the multi-criteria norm, which has to be maximized as well.

In general, the prediction quality has a fractal and even incomplete variation, since the time series could be unidentifiable for certain combinations of structural indices. Figure 2.22 displays a specific example of PQ variation for ARMA models associated to scalar time series (i.e. on one measure channel).

Two structural indices define the axes of this variation, namely: the polynomial degree of the AR part (na) and the polynomial degree of the MA part (nc). The global maximum point, $(na=19, nc=8)$, practically is drowned in an agitated sea, full of waves and, moreover, is located in the immediate neighborhood of a very sharp minimum point.

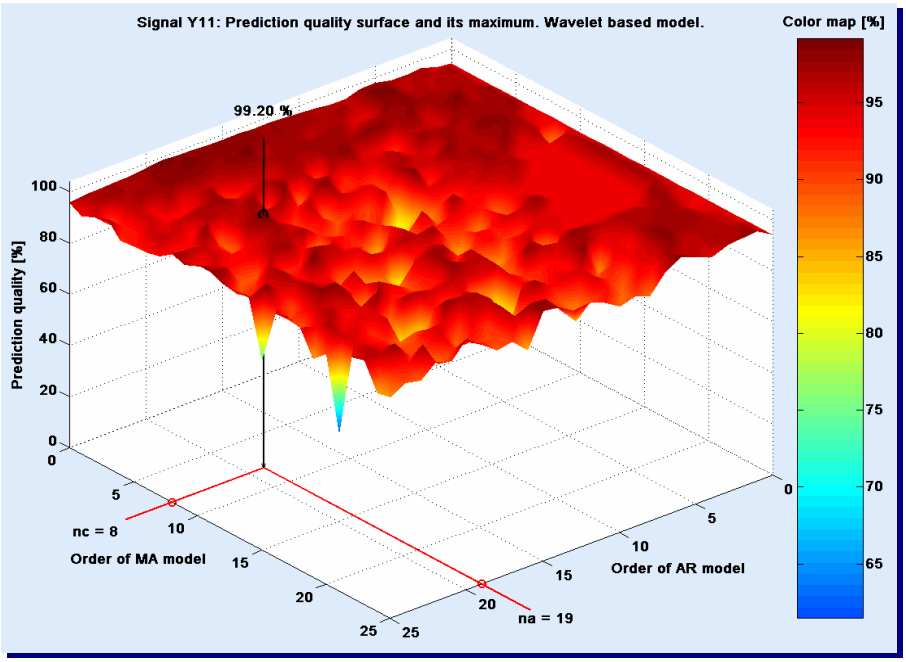


Figure 2.22. *Example of PQ fractal variation.*

By using global metaheuristics, an optimization problem such as the multivariable prediction can be solved in reasonable time. In our application, a version of Algorithm 2.9 (APSOA) was implemented in view of optimum predictor selection.

The application refers to monitoring and prediction of greenhouse ecological parameters set, for automatic control purpose. The monitored plants and the related ecological parameters are shown in Figure 2.23.

In Figure 2.24, the variations of some acquired signals are drawn, namely from the plant no. 5 (where 6 over the 7 ecological parameters have been monitored).

Some wireless sensors were used to complete the time series acquisition. The acronyms of the 7 measured parameters are explained in Table 2.2.

The Figure 2.24 reveals several insights. Firstly, one can note the existing (and expected) correlations between the soil and air temperature or between air humidity and dew point. Secondly, the leaf wetness constitutes an indicator of plant comfort level, correlated (although less obviously) to its soil moisture level. Forecasting each measurement channel, without taking into account these correlations, leads to a decreased prediction quality. According to this remark, in order to reduce the computational burden, the six parameters can be grouped both into 3 pairs or in a block of 4 and a pair.

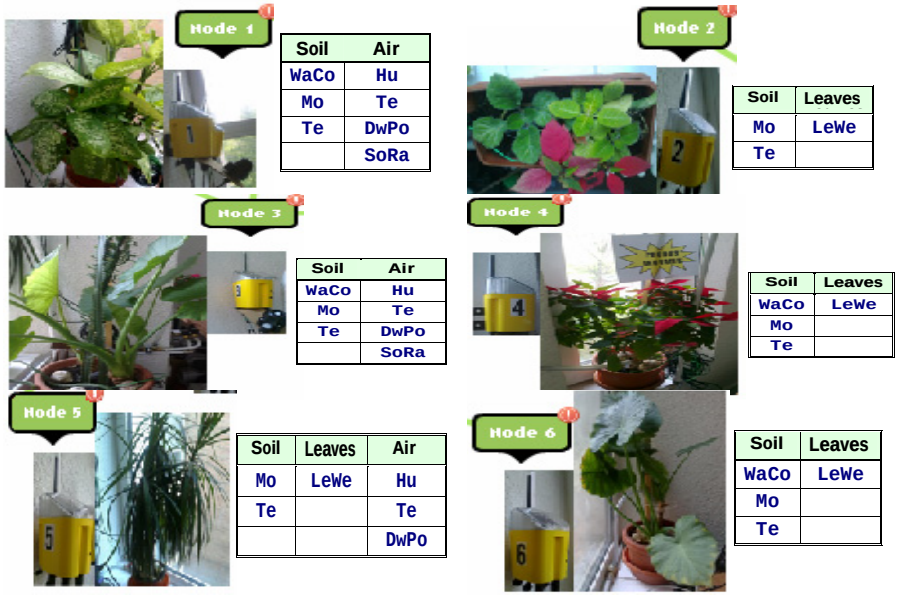


Figure 2.23. *Plants in a greenhouse and their ecological parameters.*

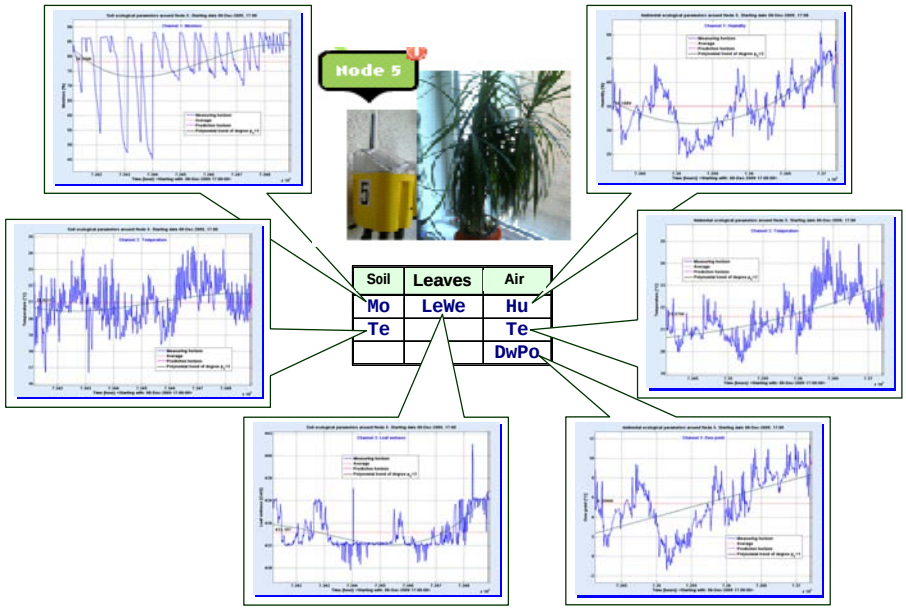


Figure 2.24. *Variations of the ecological parameters coming from plant #5 in the greenhouse.*

Table 2.2. *List of monitored ecological parameter from a greenhouse.*

Acronym	Parameter
DwPo	*Dew Point*
Hu	*Humidity*
LeWe	*Leaf Wetness*
Mo	*Moisture*
SoRa	*Solar Radiation*
Te	*Temperature*
WaCo	*Water Content*

Another important remark is concerned with the noise level of each signal. The temperature parameter seems to be the most affected by the noise. But for the other parameters, the SNR is satisfactory (i.e. quite large). Therefore, one expects that the temperatures are predicted with a lower accuracy, compared to the other parameters.

The tested optimal predictors were chosen from two aforementioned categories, by using an APSOA. The search space includes more than 3×10^{12} predictive models to be tested. In this context, a particle represents a vector that contains all the unknown structural indices of the identification model. Three types of predictors were considered: ARMA, ARMAX and Kalman-Bucy. The particle populations associated to the 3 predictors are shown in Figure 2.25. The particle values are non-negative integers.

The first type of predictors (PARMA) is associated to a relatively simple population (planar). Its purpose is to find the best prediction for each measure channel, despite the correlations between the channels. For the second type of predictors (PARMAX), the population of particles is a three-dimension block, due to its high complexity. The particle of the population enhanced, in order to take into account the structural indices of all the measure channels. A similar structure, but slightly simpler, describes the population of the third type of predictors (KARMA). The APSOA was practically implemented in 3 versions, each of each being adapted to a type of population.

The best performance of the above evolutionary strategies has to be evaluated against two parameters: speed and accuracy (or quality) of prediction.

Depending on the algorithm speed, the ranking is the following (as expected): PARMA is the fastest (being the less complex one), then KARMA takes the second place (with medium complexity) and, finally, PARMAX is the slowest (as the most complex one). More specifically, PARMA needs a few tens of minutes to perform a prediction (on a non-parallel machine with regular performance), while PARMAX can last up to 6 hours. However, this duration is not too large, since the sampling period of the greenhouse signals varies from 4 to 6 hours. (In general, the ecological signals have a fairly slow variation.)

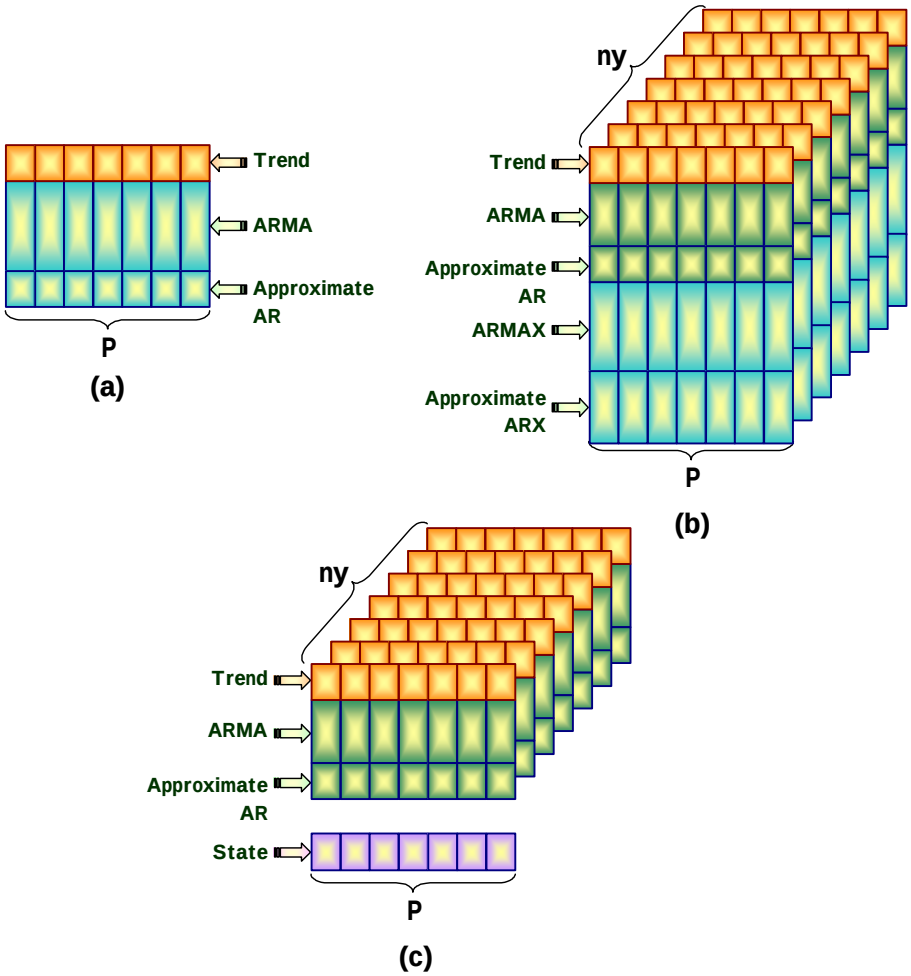


Figure 2.25. *Particle populations for the predictors: (a) ARMA, (b) ARMAX, (c) Kalman-Bucy.*

In terms of accuracy, the ranking is opposite: PARMAX dominates the scene with PQ over 80% in all cases. It is followed by PARMA and KARMA, which seem to share the places 2 and 3, depending on the data block. Their PQ ranges between 55% and 80%.

If rapid and sufficiently accurate prediction is required, then PARMA (in a version that accounts for the correlation between the measure channels) probably constitutes the best predictor that achieves this trade-off.

In order to complete this analysis, return to the greenhouse. In Figure 2.26, the *suffering level* of plants before and after installing the predictive control system is revealed. (The *suffering level* is opposite to the *satisfaction level*, as measured by the leaf wetness.)

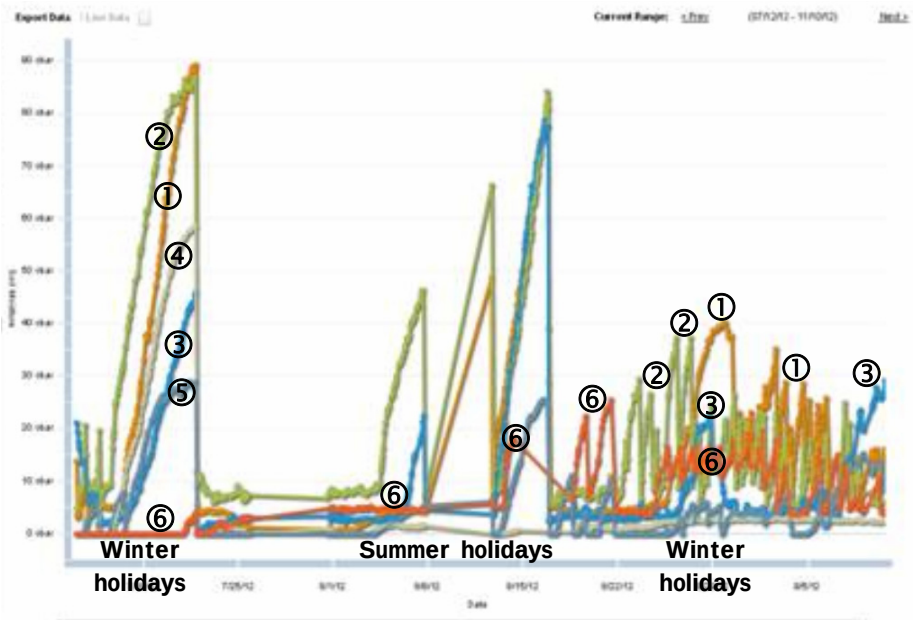


Figure 2.26. *Variations of plants suffering level in a greenhouse, before and after applying predictive control techniques.*

The variations above were plotted over one-year period, for the 6 plants. As the plants belong to different species groups, their biological rhythm is different. For example, the 6th plant supports very well the drought, while the first two are much affected by the lack of water. The greenhouse is located in a university. Before installing the automatic irrigation system based on the predictive control, one could notice on the left side of the figure, how the suffering level rapidly increases during the school holidays. Just after returning to school (on the fall semester) or during the summer randomly manual watering the plants satisfaction level suddenly improves. After installing the automatic control system, on the right side of the figure, one can observe that the plants are quite content of their life level (but at the limit) and their comfort does not change too much during the winter holidays.

The oscillations of the suffering level corresponding to plants 1, 2, 3 and 6 were determined by the following factors:

- in the beginning, the reference signals of irrigation control system were not very well calibrated; the plants were rather flooded than left without water, as

shown by the behavior of the plant #6, that more badly feels the water excess than its lack;
- the growth rhythm of each plant was not taken into account (the irrigation command was equally sent to all six plants);
- during the holidays, usually the energy supply is interrupted in the secondary areas of the university buildings (such as laboratories without continuous activity).

2.6. Optimization by harmony search

2.6.1. *Musical composition and optimization*

The idea for this metaheuristic was inspired from the techniques the musicians used in their compositions, while endeavoring to find perfect harmonies for all instruments. Early works that applied this approach to the development of an optimization metaheuristic were due to Z. Geem, J. Kim and Y. Yoon [GEE 00].

The music historians recently discovered that Mozart had composed some of his masterpieces (generally, short pieces) in unconventional manner, very little known at that time. Usually, Mozart had to compose his pieces in such a short time, that there was no time left for him to reflect on the most appropriate musical theme to the received command. For this reason, he reserved a big box in which he gradually accumulated, over the time, a huge amount of notes containing small harmonies to be combined at will. The number of these harmonies was so great that he could not make an exhaustive selection. Therefore, he used a trick. First, he randomly displayed the harmonies like in a matrix, over a large surface, such as a room floor. Then, with a set of dice, he randomly selected the elements of this virtual matrix. If the chosen harmony was not appropriate, either he would have put it back in the matrix, or he would have found a satisfactory place for it in the composition or he would have composed a new improvised harmony, close to the selected one. In the latter case, the new harmony was written on a new piece of paper and added to the others in the big box. The most Mozart's compositions are considered masterpieces of a genius though (i.e. "optimal" in the artistic sense). One could remark that he already used a Monte-Carlo technique for composing them, long time before this technique was discovered.

The metaheuristic described in this last section of the chapter follows the Mozart's approach.

The analogy between the musical composing process and the optimization process can be summarized in Table 2.3.

Table 2.3. *Analogy between musical composition and optimization.*

Musical composing process	Optimization process
Musical instrument	Decision variable
Tone	Value of the decision variable
Musical improvisation	Iteration
Musical harmony	Fitness
Quality of the musical harmony	Fitness value

2.6.2. *Harmony search model*

This metaheuristic is well suited for optimization problems with constraints, such as [2.81]. Eventually, part (or even all) of the constraints can be removed, provided that the search domain $\mathcal{S}$ is defined clearly.

The *decision variables* $\left\{\mathscr{I}_p\right\}_{p\in\overline{1,P}}$, with $P\in\mathbb{N}^*$ values, are chosen as *instruments*, in order to interpret the musical composition. These variables take values in the "tones domain" $\mathcal{S}$.

In order to start the composition process, one can randomly choose P tones of the domain $\mathcal{S}$, say $\left\{\mathscr{I}_p[0]=\mathbf{x}_p^0\right\}_{p\in\overline{1,P}}\subset\mathcal{S}$. These tones are arranged in a matrix, where their components are written on rows:

$$\mathbb{M}_H^0=\begin{bmatrix} x_{1,1}^0 & x_{1,2}^0 & \cdots & x_{1,nx-1}^0 & x_{1,nx}^0 \\ x_{2,1}^0 & x_{2,2}^0 & \cdots & x_{2,nx-1}^0 & x_{2,nx}^0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{P-1,1}^0 & x_{P-1,2}^0 & \cdots & x_{P-1,nx-1}^0 & x_{P-1,nx}^0 \\ x_{P,1}^0 & x_{P,2}^0 & \cdots & x_{P,nx-1}^0 & x_{P,nx}^0 \end{bmatrix}\begin{matrix} \Rightarrow f\left(\mathbf{x}_1^0\right) \\ \Rightarrow f\left(\mathbf{x}_2^0\right) \\ \vdots \\ \Rightarrow f\left(\mathbf{x}_{P-1}^0\right) \\ \Rightarrow f\left(\mathbf{x}_P^0\right) \end{matrix}. \qquad [2.97]$$

The matrix [2.97] stands for the *harmonic memory*. It produces a sequence of musical harmonies, $\left\{f\left(\mathbf{x}_p^0\right)\right\}_{p\in\overline{1,P}}$ as shown on the right side.

Moving towards the optimum of criterion f means improvising new tones, which are more or less related to the harmonic memory. This memory can change, depending on the new tones to make an improvisation. In general, the matrix adapts itself whenever the new tones produce harmonies with superior quality, when comparing to the existing ones.

Denote by $k \in \mathbb{N}^*$ the current iteration index of the optimization procedure. The goal of this process is to improve the harmony quality of the current memory $\mathbb{M}_H^k$, by musical improvisation. This quality is given by the best value of its harmonies, i.e. is the solution of the following optimization problem:

$$\mathbf{x}_{\mathbb{M}_H}^{\text{opt}} = \underset{p \in \overline{1,P}}{\text{argopt}} \left\{ f\left(\mathbf{x}_p^k\right) \right\}. \qquad [2.98]$$

The improvisation consists of chosing a new tone $\mathbf{y} \in \boldsymbol{S}$, by using the following rules, as inspired from the musical composition process:

- the tone has to be chosen at random, either from the harmonic memory or the tone domain;
- the selected tone can be adjusted in a certain way;
- if the selected and eventually adjusted tone is superior in harmonic quality to the *least harmonic* tone of the memory (i.e. having the lowest harmonic quality compared to the other tones of the memory), then it can replace the latter in the memory.

A tone selection is performed according to a probability $P_{\mathbb{M}} \in (0,1)$, which sets a threshold regarding how permissive is the harmonic memory to improvisations. Thus, selected tones outside the memory lines (but from the tones domain $\boldsymbol{S}$) are accepted with probability $1 - P_{\mathbb{M}}$, while existing tones in harmonic memory are selected with probability $P_{\mathbb{M}}$. In the latter case, the tone is coming from a randomly chosen row of matrix $\mathbb{M}_H^k$.

Let $\mathbf{y}$ the selected tone. Then each one of its components can be adjusted with the probability $P_{\mathbb{M}} P_a \in \left(0, P_{\mathbb{M}}\right]$, by slightly varying its value, as follows:

$$y_i \leftarrow y_i + \xi_i \Delta\omega_i, \quad \forall i \in \overline{1,nx}, \qquad [2.99]$$

where $\xi_i \in [-1,+1]$ is randomly selected and expresses the "composer's inspiration", while $\Delta\omega_i > 0$ is associated to the maximum bandwidth allowing the tone frequency to change. In general, the bandwidths $\left\{\Delta\omega_i\right\}_{i \in \overline{1,nx}}$ are a priori fixed, depending on the representation scale of tones domain. For example, if the hyper-parallelepiped envelope of this domain is known (like in the case of BeeA), then the bandwidth could be defined as follows:

$$\Delta\omega_i = \frac{x_i^{\max} - x_i^{\min}}{2}, \quad \forall i \in \overline{1,nx}. \qquad [2.100]$$

The current component y_i could be kept unchanged with probability $1-P_{\mathbb{M}}P_a$. But if all the components are not modified and the tone comes from the $\mathbb{M}_H^k$ memory, then it is better to replace it by a combination of tones in this memory. For example, a possible approach is to take each tone in $\mathbb{M}_H^k$ with its own harmonic quality and to compute the weighted average of the whole set:

$$\mathbf{y} = \frac{\sum_{p=1}^{P} f\left(\mathbf{x}_p^k\right)\mathbf{x}_p^k}{\sum_{p=1}^{P} f\left(\mathbf{x}_p^k\right)}. \qquad [2.101]$$

If the product between $P_{\mathbb{M}}$ and P_a probabilities is too small, then, during the composition process, $\mathbf{y}$ will often be selected among the memory tones. This can slow down the search for the optimum or, even worse, can trap the search into a local optimum.

Anyway, before continuing with the new tone $\mathbf{y}$, one has to be sure it is viable. Otherwise it has to be made viable by a specific technique, as earlier described in this chapter.

The new (viable) tone replaces the worst row of the harmonic memory (actually hosting the *dissonant* tone), if its harmonic quality is superior. Thus, renewal of the harmonic matrix is performed as below (where $q \in \overline{1,P}$ indicates the position of the dissonant tone):

$$\mathbb{M}_H^{k+1} = \begin{bmatrix} \mathbf{x}_1^k \\ \mathbf{x}_2^k \\ \vdots \\ \mathbf{y} \rightarrow \mathbf{x}_q^k \rightarrow \\ \vdots \\ \mathbf{x}_{P-1}^k \\ \mathbf{x}_P^k \end{bmatrix}. \qquad [2.102]$$

If the quality of new tone is lower than the quality of the harmonic matrix components, then the matrix is passed unchanged to the next iteration:

$$\mathbb{M}_H^{k+1} = \mathbb{M}_H^k. \qquad [2.103]$$

The composition process (in fact the optimization procedure) stops either after operating a certain number of improvisations (say $K_{\max}$) or after noticing that the harmonic memory remained unchanged for more than a specified number of iterations (say M_s). The solution to the optimization problem is given by the highest quality tone that belongs to the harmonic memory.

2.6.3. *Standard harmony search algorithm*

The standard optimization procedure related to the harmonic model is summarized in Algorithm 2.13.

Algorithm 2.13. *Standard optimization procedure by harmony search.*

1. Input data:

- Optimization criterion, f, to be minimized or maximized (see, for example, the problem [2.81]).
- Constraint expressions, $\{g_l\}_{l\in\overline{1,L}}$ and $\{h_r\}_{r\in\overline{1,R}}$ (if any).
- Optionally: envelope of the search space, $\mathcal{S}\subseteq\mathbb{R}^{nx}$, seen as a hyper-parallelepiped, i.e. the limits: $\{x_i^{\min}\}_{i\in\overline{1,nx}}$ and $\{x_i^{\max}\}_{i\in\overline{1,nx}}$.
- Configuring parameters:
  - the number of instruments in the orchestra, $P\in\mathbb{N}^*$ (usually, of few tens);
  - the probability showing how permissive is the harmonic memory to improvisations, $P_{\mathbb{M}}\in(0,1)$;
  - the (conditional) probability of the tone adjustment, $P_a\in(0,1)$;
  - if possible, the maximum frequency bandwidths that allow the tones to be adjusted, $\{\Delta\omega_i\}_{i\in\overline{1,nx}}$;
  - the minimum resolution of roulette within BGA, when employed to generate PRS, $N\in\mathbb{N}^*$ (by default: $N=1000$, i.e. the selection set of type $[a,b]\subset\mathbb{R}$ has to include at least 1000 uniformly distributed points);
  - the maximum number of improvisations, $K_{\max}\in\mathbb{N}^*$;
  - the survival factor of the harmonic memory, $M_s\in\mathbb{N}^*$ (by default: $M_s=0.1\cdot K_{\max}$).

2. Initialization

a. If the maximum frequency bandwidths are not defined, evaluate them according to the tone domain topology. For example, if the rectangular envelope of this domain is defined, one can use the definition [2.100] in this aim:

$$\Delta\omega_i = \frac{x_i^{\max} - x_i^{\min}}{2}, \quad \forall i \in \overline{1,nx}.$$

b. Choose the first P tones from the search domain, $\left\{\mathbf{x}_p^0\right\}_{p\in\overline{1,P}}$, in order to build the first harmonic memory, $\mathbb{M}_H^0$, by means of definition [2.97]. (If no preferred technique exists, use a U-PRSG in this aim.)

c. Evaluate the best harmonic quality of the tones in memory $\mathbb{M}_H^0$:

$$\left(\mathbf{x}_{\mathbb{M}_H}^{\text{opt}}, f_{\mathbb{M}_H}^{\text{opt}}\right) = \underset{p\in\overline{1,P}}{\text{opt}}\left\{f\left(\mathbf{x}_p^0\right)\right\}.$$

d. Evaluate the worse harmonic quality of the tones in memory $\mathbb{M}_H^0$:

$$\left(\mathbf{x}_{\mathbb{M}_H,q}^{\text{tpo}}, f_{\mathbb{M}_H,q}^{\text{tpo}}\right) = \underset{p\in\overline{1,P}}{\text{tpo}}\left\{f\left(\mathbf{x}_p^0\right)\right\},$$

With the natural notations (as employed in case of APSOA). Here, $q \in \overline{1,P}$ indicates the position of the *dissonant* tone (with the minimum harmonic quality).

e. Initialize the survival index of harmonic memory, $m = 0$.

3. For $k \in \overline{0,K_{\max}-1}$ (where k is the iterations index) and while $m \le M_s$, do:

3.1. Use a U-PRSG of resolution at least equal to N, in order to select a number $\tau \in [0,1]$.

3.2. If $\tau \le P_{\mathbb{M}}$, then perform an improvisation starting from the existing tones of harmonic memory $\mathbb{M}_H^k$. Use a U-PRSG in order to choose a number between 1 and P, for example p. The new tone origin is then $\mathbf{y} = \mathbf{x}_p^k$.

3.3. Otherwise, the improvisation is performed by selecting any tone $\mathbf{y} \in \mathcal{S} \setminus \mathbb{M}_H^k$ (outside the harmonic memory), with an adapted U-PRSG.

3.4. Use a U-PRSG of nx size and resolution at least equal to N, in order to choose the set the numbers $\left\{\tau_i\right\}_{i\in\overline{1,nx}} \subset [0,1]$ that decide which tone component from $\mathbf{y}$ has to be adjusted.

3.5. For $i \in \overline{1,nx}$:

3.5.1. If $\tau_i < P_{\mathbb{M}} P_a$, then the component y_i has to be adjusted. To do so:

3.5.1.1. Use a U-PRSG of nx size and resolution at least equal to N , in order to choose the composer's inspiration factor $\xi \in [-1,+1]$.

3.5.1.2. Adjust the component, by means of equation [2.99]:

$$y_i \leftarrow y_i + \xi \Delta\omega_i \, .$$

3.5.2. Otherwise, preserve the component y_i .

3.6. Make viable the new selected or eventually adjusted tone $\mathbf{y}$, according to the search space topology.

3.7. If despite of all, $\mathbf{y}$ still belongs to the tones of the current harmonic memory, i.e. if $\mathbf{y} \in \mathbb{M}_H^k$, then it is better to change it. For example, one can use the definition [2.101] in this aim:

$$\mathbf{y} = \frac{\sum_{p=1}^{P} f\left(\mathbf{x}_p^k\right) \mathbf{x}_p^k}{\sum_{p=1}^{P} f\left(\mathbf{x}_p^k\right)} \, .$$

Make this tone viable, if necessary.

3.8. If the harmonic quality of the new tone is good enough, i.e. if $f(\mathbf{y})$ is better than $f_{\mathbb{M}_H,q}^{\text{tpo}}$, then this tone should replace the dissonant tone of the matrix, located in position q . More specifically:

3.8.1. Apply the operation suggested by [2.102]. Thus, $\mathbf{x}_q^{k+1} = \mathbf{y}$.

3.8.2. Update the most harmonious tone and its quality. If $f(\mathbf{y})$ is better than $f_{\mathbb{M}_H}^{\text{opt}}$, then: $\mathbf{x}_{\mathbb{M}_H}^{\text{opt}} \leftarrow \mathbf{y}$ and $f_{\mathbb{M}_H}^{\text{opt}} \leftarrow f(\mathbf{y})$. Otherwise, the most harmonious tone in the composition remains unchanged.

3.8.3. Update the dissonant tone and its quality:

$$\left(\mathbf{x}_{\mathbb{M}_H,q}^{\text{tpo}}, f_{\mathbb{M}_H,q}^{\text{tpo}}\right) = \underset{p \in \overline{1,P}}{\text{tpo}} \left\{ f\left(\mathbf{x}_p^{k+1}\right) \right\} .$$

3.8.4. Since the harmonic memory has changed, update the survival index: $m = 0$.

3.9. Otherwise, the harmonic matrix remains unchanged: $\mathbb{M}_H^{k+1} = \mathbb{M}_H^k$. In this case, increment the survival index: $m \leftarrow m+1$.

4. Return:

- The optimal solution of harmonic memory (provided by the most harmonious tone): $\left(\mathbf{x}_{\mathbb{M}_H}^{\text{opt}}, f_{\mathbb{M}_H}^{\text{opt}}\right)$.

This *Harmony Search Algorithm* (HSA) is simpler than other global heuristic procedures. Its similarity with Monte-Carlo Method is obvious. But this does not means the HSA is not a good optimization metaheuristic. On he contrary, it proved to be quite effective, especially in problems where the constraints are not too restrictive. The user keeps a reasonable control of the exploration-exploitation trade-off, by properly selecting the probabilities $P_{\mathbb{M}}$ and P_a (possibly by learning, after several attempts).

The optimization by harmony search is comparable to the evolutionary algorithms in terms of accuracy. In many applications, it provides close results with less computational burden.

This approach has many applications, for example, as reported in the following publications: [GEE 02], [AYA 10], [ERD 08], [BET 08].

2.6.4. *Application example*

Assume the problem is to find the solution $\mathbf{x} = \left[x_1 \;\; x_2\right]^T$ corresponding to the minimum of $f(\mathbf{x}) = 5x_1^2 - 9x_1x_2 + 5x_2^2$, with the constraint $g(x) = 25 - 16x_1x_2 \le 0$, given that the values of x_1 and x_2 belong to the set $\overline{0.5:0.5:10}$ (i.e. the interval [0.5,10] sampled by 0.5 period). The optimal solution obtained by starting from the Karush-Kuhn-Tucker conditions [BOR 13] is: $\mathbf{x} = \left[1.5 \;\; 1.5\right]^T$.

The parameters initially are set to: $P = 10$, $P_{\mathbb{M}} = 0.72$ and $P_a = 0.18$.

The initial harmonic memory corresponds to Table 2.4. The dissonant tone is the 10^{th} one, while the most harmonious tone is the first one.

Table 2.4. *Example of initial harmonic memory.*

Instrument no.	x_1	x_2	$f(\mathbf{x})$
1	**3.0**	**4.5**	**24.75**
2	2	4	28
3	7.5	6	56.25
4	4.5	7	62.75
5	5.5	8	75.25
6	8.5	5.5	91.75
7	10	9	95
8	10	10	100
9	5.5	1	106.7
10	1	9	329

The first 13 iterations were rejected as invalid or not bringing improvements. The 14th iteration allows replacing the dissonant tone (the 10th in the harmonic memory), as shown in Table 2.5. Beside the dissonant tone, other tones have been replaced as well. But the most harmonious tone was not affected.

Table 2.5. *Example of harmonic memory improvement.*

Instrument no.	x_1	x_2	$f(\mathbf{x})$
1	**3**	**4.5**	**24.75**
2	2	4	28
3	7.5	6	56.25
4	4.5	7	62.75
5	5.5	8	75.25
6	8.5	5.5	91.75
7	10	9	95
8	6	9	99
9	10	10	100
10	5.5	1	106.75

Finally, the optimal solution is obtained at the 50th iteration, the final harmonic memory being described in Table 2.6. It corresponds indeed to Karush-Kuhn-Tucker's solution.

Table 2.6. *Example of optimal harmonic memory.*

Instrument no.	x_1	x_2	$f(\mathbf{x})$
1	**1.5**	**1.5**	**2.25**
2	2	1.5	4.25
3	2.5	2.5	6.25
4	3	2.5	8.75
5	3	3	9
6	2	3	11
7	3	3.5	11.75
8	3.5	3.5	12.25
9	2.5	3.5	13.75
10	3.5	4	15.25

Bibliography

[ABD 12] ABDULLAH A., DERIS S., MOHAMAD M.S., HASHIM S.Z.M., "A new hybrid firefly algorithm for complex and nonlinear problem", *Distributed Computing and Artificial Intelligence – Advances in Intelligent and Soft Computing*, vol. 151, pp. 673-680, 2012.

[ALF 11] ALFI A., KHOSRAVI A., RAZAVI S.E., "Bee Algoritm–Based Nolinear Optimal Control Applied To A Continuous Stirred-Tank Chemical Reactor", *Global Journal of Pure & Applied Science and Technology - GJPAST*, vol. 1, no. 2, pp. 73-79, 2011.

[AST 94] Aström K.J., Nilsson J., "Analysis of a Scheme for Iterated Identification and Control", *IFAC Symposium on System Identification, SYSID'94*, Copenhagen, Denmark, 1994.

[AUN 11] AUNGKULANON P., CHAI N., LUANGPAIBOON P., "Simulated Manufacturing Process Improvement via Particle Swarm Optimisation and Firefly Algorithms", *Proceedings of International Multiconference of Engineers and Computer Scientists*, vol. 2, pp. 1123-1128, 2011.

[AYA 00] AYACHI I., KAMMARTI R., KSOURI M., BORNE P., "Harmony search algorithm for the container storage problem", *Proceedings of MOSIM'10 International Conference, Hammamet, Tunis*, 2010.

[BAE 00] BAECK T., FOGEL D.B., MICHALEWICZ Z., *Evolutionary Computation – Basic and Advanced Algorithms and Operators* (2 volumes), Institut of Physics Publishing House, 2000.

[BAH 08] BAHAMISH H.A.A., ABDULLAH R., SALAM R.A., "Protein Conformational Search Using Bees Algorithm", *Second Asia International Conference on Modeling & Simulation (AICMS 08)*, Kuala Lumpur, Malaysia, IEEE Press, pp. 911-916, 2008.

[BAK 85] BAKER J.E. – "Adaptive Selection Methods for Genetic Algorithms", *Proceedings of the first International Conference on Genetic Algorithms and Applications, Erlbaum Printing House, Ed. J.J. Grefenstette*, 1985.

[BAK 87] BAKER J.E. – "Reducing Bias and Inefficiency in the Selection Algorithm", *Proceedings of the second International Conference on Genetic Algorithms and Applications, Erlbaum Printing House, Ed. J.J. Grefenstette*, 1987.

[BAL 13] BALEZENTIS T., BALEZENTIS A., "Survey on Development and Applications of the Multi-criteria Decision Making Metod MULTIMORA", *Journal of Multi-criteria Decision Analysis*, DOI b10.102/mcda 1501, 2013.

[BAN 90] BANA & COSTA (ed.), *Readings in Multiple Criteria Decision Aid*, Springer Verlag, 1990.

[BAR 91] BARON C., GOMEZ S., *The exponential tunneling method*, Report, IIMAS 1, National Autonomous University of Mexico 3, pp. 1-23, 1991.

[BAR 10] BARROS G., "Herbert A. Simon and the Concept of Rationality. Boundaries and Problems", *Brazilian Journal of Political Economy*, vol. 30, no. 3, pp. 455-472, 2010.

[BET 08] BETAR AL M., KHODER A., GANI T. "A harmony search algorithm for the university course timetabling", *Proceedings of the 7-th International Conference on Practice and Theory of Timetabling, PATAT-2008, Montreal, Canada*, 2008.

[BOE 98] BOER DE L., WEGEN VAN DER L., TELGEN J., "Outranking Methods to Support Supplier Selection", European Journal of Purchasing and Supply Management, no. 4, pp. 119-118, 1998.

[BOL 78] BOLTIANSKI V.C., *Optimal Control of Discrete Systems*, John Wiley and Sons, 1978.

[BON 84] BONCZEK R.H., HOLSAPPLE C. W., WHINSTON A.B., *Foundations of Decision Support Systems*, Academic Press, New York, U.S.A., 1984.

[BON 99] BONABEAU E., DORIGO M., THERAULAZ G., *Swarm Intelligence: From Natural to Artificial Systems*, Oxford University Press, U.K., 1999.

[BOR 53] BORDA J., "Mémoire sur les élections au scrutin", *Histoire de l'Académie Royale des Sciences*, Année MDCCLXXXI, pp. 657-665. ("Mathematical Derivation of an Election System"), *ISIS*, vol 44, no.1-2, pp. 42-51, 1784), 1953.

[BOR 10] BORNE P., FILIP F.G. , BENREJEB M., POPESCU D., *Automatique Avancée et Informatique Appliquée*, Éditions de l'Académie Roumaine, Bucarest, Roumanie, 2011.

[BOR 13] BORNE P., POPESCU D., FILIP GH.F., STEFANOIU D., *Optimization in Engineering Sciences – Exact Methods*, ISTE & John Wiley and Sons, London, U.K., 2013.

[BOU 06] BOUYSSOU D., DUBRIS D., PIRLOT M., PRADE H., *Concepts et méthodes pour aide à la décision*, vol. 3 : "Analyse multicritère", Hermes-Lavoisier, Paris, France, 2006.

[BRA 82] BRANS J. P. "L'Inginerie de la décision: Elaboration d'instruments d'aide à la décision. La méthode PROMETHE", in *L'aide à la décision : nature, instruments et perspectives d'avenir* (Nadeau, R., Landry, M. eds.), Presse de l'Université Laval, Québec, 1982.

[BRA 05] BRANS J.P., MARECHAL B., "PROMETHEE Methods", chapter in [FIG 05a], pp. 163-194, 2005,

[CAL 79] CALIN S., TERTISCO M., DUMITRACHE I., POPEEA C., POPESCU D., *Optimisation en Automatisation Industrielle*, Editura Tehnica, Bucarest, Roumanie, 1979 (en roumain).

[CHA 61] CHARNES A., COOPER W.W., *Management Models and Industrial Application of Linear Programming*, J. Wiley and Sons, New York, 1961.

[CHA 77] CHARNES A., COOPER W.W., "Goal Programming and Multiple Objective Optimization. Part 1", *European Journal of Operational Research*, vol. 1, no. 39, 1977.

[CHA 11] CHAI N., AUNGKULANON P., LUANGPAIBOON P., "Bees and Firefly Algorithms for Noisy Non-Linear Optimisation Problems", *Proceedings of International Multiconference of Engineers and Computer Scientists*, vol. 2, pp. 1449–1454, 2011.

[CHA 12] CHATTERJEE A., MAHANTI G.K., "Design of a Fully Digital Controlled Reconfigurable Switched Beam Concentric Ring Array Antenna using Firefly and Particle Swarm Optimization Algorithm", *Progress in Elelectromagnetic Research B*, vol. 36, pp. 113-131, 2012.

[CHO 53] CHOQUET G., "Theory of capacities", *Annales de l'Institut Fourrier*, vol. 5, pp. 131-295, 1953.

[CIV 13] CIVS-Condorcet Internet Service (www.civs.cs.cornell, accessed on 05.01.2014).

[CLE 96] CLEMEN R.T., *Making Hard Decisions. An Introduction to Decision Analysis*. 2nd edition, Duxbury Press, Belmont, U.S.A., 1996.

[CLE 14] CLEMEN R.T., *Making Hard Decision Tools*. 3rd edition, Terence Reilly Babson College, 2014.

[COC 73] COCHRANE J.I., ZELENY M., *Multiple Criteria Decision Making*. University of South Carolina Press, 1973.

[COH 95] COHEN L., *Time-Frequency Analysis*, Prentice Hall, New Jersey, USA, 1995.

[COL 92] COLORNI A., DORIGO M., MANIEZZO V., "Distributed Optimization by Ant Colonies", *Proceedings of ECAL'91 conference, Elsevier Publishing, Paris, France*, pp. 134-142, 1992.

[COL 03] COLLETTE Y., SIARRY P., *Optimisation multiobjectif*, Eyrolles, Paris, France, 2003.

[CON 85] CONDORCET MARQUIS DE, *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*, L'Imprimerie Royale, Paris, France, 1785.

[CUL 11] CULITA J., STEFANOIU D., DUMITRASCU A., "EcoMonFor – A System for Greenhouses Monitoring and Forecasting", *Industrial Simulation Conference ISC-2011*, Venice, Italy, pp. 262-269, June 6–8, 2011.

[DAR 59] DARWIN C.R., *On the Origin of Species by Means of Natural Selection*, London Press, U.K., 1859.

[DAR 71] DARWIN C.R., *The Descent of Man and Selection in Relation to Sex*, London Press, U.K., 1871.

[DAU 92] DAUBECHIES I., *Ten Lectures on Wavelets*, CBMS Lecture Notes, SIAM, No. 61, 1992.

[DEK 12] DEKHICI L., BELKADI K., BORNE P., "Firefly algorithm for economic power dispatching with polluants emission", *Informatica Economica*, vol. 16, no. 2, pp. 45-57, 2012.

[DIO 97] DION J.M., POPESCU D., *Optimisation des Systèmes. Commande Optimale*, Diderot, Paris, 1997.

[DOR 92] DORIGO M., *Optimization, Learning and Natural Algorithms*, Thèse de Doctorat, Politecnico di Milano, Italie, 1992.

[DOR 96] DORIGO M., MANIEZZO V., COLORNI A., "The Ant System: Optimization by a Colony of Cooperating Agents", *IEEE Transactions on Systems, Man, Cybernetics*, Part B, no. 26, pp. 29-41, 1996.

[DOR 97] DORIGO M., GAMBARDELLA L.M., "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem", *IEEE Transactions on Evolutionary Computing*, vol. 1, no.1, pp. 53-66, 1997.

[DOR 99] DORIGO M., DI CARO G., GAMBARDELLA L.M., "Ant Algorithms for Discrete Optimization", *Artificial Life*, vol. 5, no.2, pp. 137-172, 1999.

[DOR 04] DORIGO M., STÜTZLE T., *Ant Colony Optimization*, MIT Press, Cambridge, USA, 2004.

[DRE 05] DRÉO J., PÉTROWSKI A., SIARRY P., TAILLARD E., *Métaheuristiques pour l'optimisation difficile*, Éditions Eyrolles, 2005.

[DYE 92] DYER J.S., FISHBURN P.C., STEUER R.E., WALLENIUS J.S., ZIONTS S., "Multiple Criteria Decision Making, Multiattribute Utility Theory: The Next Ten Years", *Management Science*, vol. 38, no. 5., pp. 645-654, 1992.

[EBE 01] EBERHART R., KENNEDY J., SHI Y., "Swarm Intelligence", chapter in *Evolutionary Computation*, Morgan Kaufman, 2001.

[ECK 87] ECKHARDT R., "Stan Ulam, John von Neumann, and the Monte Carlo method", *Los Alamos Science*, special issue no. 15, pp. 131-137, 1987.

[EDW 94] EDWARDS W. BARRON F.H., "SMARTS AND SMARTER: Improved Simple Method for Multiattribute Utility Measurement", *Organizational Behaviour and Human Decision Processes*, no. 60, pp. 306-325, 1994.

[EHR 10] EHRGOTT M., FIGUIERA R.J., GRECO S., *Trends in Multicriteria Decision Analysis.* Springer, New York, Dordrecht, Heidelberg, London, 2010.

[ENN 04] ENNIGRON M., GHÉDIRA K., "Flexible Job-Shop Scheduling with Multi-Agent System and Taboo Search", *Journal Européen des Systèmes Automatisés JESA*, vol. 38, no. 7-8, 2004.

[ERD 08] ERDAL F., SOKA M., "Effect of beam spacing in the harmony search bases optimum design of grillages", *Asian Journal of Civil Engineering (Building and Housing)*, vol. 9, no. 3, pp. 215-228, 2008.

[FAH 12] FAHMY A.A., KALYONCU M., CASTELLANI M., "Automatic Design of Control Systems for Robot Manipulators Using the Bees Algorithm" (Part I), *Proceedings of the Institution of Mechanical Engineers*: *Journal of Systems and Control Engineering*, vol. 226, no. 4, pp. 497-508, 2012.

[FAR 12] FARAHANI S.M., ABSHOURI A.A. , NASIRI B., MEYBODI M.R., "Some hybrid models to improve firefly algorithm performance", *International Journal of Artificial Intelligence*, vol. 8, no. 12, pp. 97-117, 2012.

[FEO 95] FEO T., RESENDE M., "Greedy randomized adaptive search procedure", *Journal of Global Optimization*, vol. 2, pp. 860-878, 1995.

[FIG 05a] FIGUEIRA J.S., GRECO S., EHRGOTT M., *Multiple Criteria Decision Analysis ; State of the Art Surveys*, Springer Science and Business Media, New York, U.S.A., 2005.

[FIG 05b] FIGUEIRA J., GRECO S., EHRGOTT M., "Introduction", chapter in [FIG 05a], pp. XVII-XXX, 2005.

[FIG 05c] FIGUEIRA J, MOUSSEAU A., ROY B., "ELECTRE Methods", chapter in [FIG 05a], pp. 133-162, 2005.

[FIL 81] FILIP F.G., *Contribuţii la conducerea ierarhizată a proceselor complexe* (*Contributions to Hierarchical Control of Complex Systems*). Ph.D. Thesis, Polytechnic Institute of Bucharest, Romania, 1981 (in Romanian).

[FIL 83a] FILIP F.G., DONCIULESCU D.A., “"On an Online Direct Dynamic Coordination Method in Process Industry”. *IFAC J. Automatica*, vol. 19, no. 1, pp. 317-320, 1983.

[FIL 83b] FILIP F.G., NEAGU G., DONCIULESCU D.A., “Jobshop Scheduling Optimization in Real Time Production Control”, *Computers in Industry*, vol. 4, no. 3, pp. 395-403, 1983.

[FIL 85] FILIP F.G., DONCIULESCU D.A., GASPAR R., MURATCEA M., ORASANU L., “Multilevel optimization algorithms in Computer Aided Production Control in Process Industry”, *Computers in Industry*, vol. 6, no. 1, pp. 47-57, 1985.

[FIL 91] FILIP F.G., “System Analysis and Expert Systems Techniques for Operative Decision Making”, *Systems Analysis Modelling Simulation*, vol. 8, no. 3, pp. 203-219, 1991.

[FIL 98] FILIP F.G., “Optimization Methods with Sparse Matrices and Relatively Constant Parameters”, *Systems Analysis, Modeling and Simulation*, no. 33, pp. 407-438, 1998.

[FIL 08a] FILIP F.G., “Decision Support and Control for Large–Scale Complex Systems”, *Annual Reviews in Control*, vol. 32, no. 1, pp. 61-70, 2008.

[FIL 08b] FILIP F., POPESCU D., MATEESCU M., “Optimal Decisions for Complex Systems – Software Packages”, *Mathematics and Computers in Simulation*, no. 16, 2008.

[FIL 09] FILIP F.G., LEIVISKA K., “Large-scale complex systems”, chapter in: *Handbook of Automation* (ed. S.Y. Nof), Springer, Dordrecht, Germany, pp. 619-638, 2009.

[FIL 14] FILIP F.G., SUDUC A.-M., BIZOI M., “DSS in Numbers”, *Technological and Economic Development of Economy*, vol. 20, no. 1, pp. 154-164, 2014.

[FIS 95] FISHMAN G. S., *Monte Carlo: Concepts, Algorithms, and Applications*, Springer, New York, U.S.A., 1995.

[FOR 85] FOREST S., *Scaling Fitness in the Genetic Algorithm*, Documentation for PRISONERS DILEMMA and NORMS Programs that use the Genetic Algorithm, Preprint, 1985.

[FOU 04] FOULLOY L., POPESCU D., DAUPHIN-TANGUY G., *Modélisation, Identification et Commande des Systèmes*, Éditions de l Académie Roumaine, Bucarest, 2004.

[GAL 51] GALE D., KHUN H.W., TUCKER A.W., *Linear Programming and the Theory of Games*, John Willey and sons, New York, USA, 1951.

[GAN 12] GANG K., LU Y., PENG Y., SHI Y. “Evaluation of Classification Algorithms Using MCDM and Rank Correlation”, *International Journal of Information Technology Decision Making*, vol. 11, no. 1, pp. 197-225, 2012.

[GAR 96] GARNER B., WELZL E., *Linear Programming-Randomization and Abstract Framework*, Springer Verlag, Berlin, Germany, 1996.

[GEE 00] GEEM Z., KIM J., YOON Y. “Optimal layout of pipe networks using harmony search”, *Proceedings of the 4-th International Conference on Hydro-science and Engineering, Seoul, South Korea*, 2000.

[GEE 02] GEEM Z., KIM J., LOGANALHOM G. “Harmony search optimization: Application to pipe network design”, *International Journal of Modelling and Simulation*, vol. 22, no. 2, pp. 125-133, 2002.

[GEN 98] GENTIL S., POPESCU D., *Commande Numérique et Intelligence Artificielle en Automatique*, Editura Tehnica, Bucarest, Romania, 1998.

[GEV 95] GEVERS M., "Identification for Control", *IFAC Conference ACASP'95*, Budapest, Hungary, 1995.

[GHE 07] GHÉDIRA K., *Optimisation combinatoire par métaheuristiques*, Éditions TECHNIP, France, 2007.

[GLO 89] GLOVER F., "Taboo search – Part I", *ORSA Journal on Computing*, no. 1, pp. 190-206, 1989.

[GLO 90] GLOVER F., "Taboo search – Part II", *ORSA Journal on Computing*, no. 2, pp. 4-32, 1990.

[GOL 91] GOLDBERG D.E., DEB K., Chapitre: "A Comparative Analysis of Selection Schemes used in Genetic Algorithms", in *Foundation of Genetic Algorithms 1*, Ed. G. Rawlings & L.D. Whitley, Morgan Kaufmann Editions, USA, 1991.

[GOL 94] GOLDBERG D.E., *Algoritmes génétiques. Exploration, optimisation et apprentissage automatique*, Addison-Wesley, France, 1994.

[GOM 97] GOMES L.F.A.M., MURY A.R., GOMES C.F.S., "Multicriteria Ranking with Ordinal Data", *SAMS*, vol. 27, pp. 139-145, 1997.

[GOM 13] GOMES L.F.A. M., MACHEDO M.A.S., RANGEL L.A.D., "Behavioural multi-crieria decision analysis: the TODIM method with criteria interaction", *Ann. Oper. Res.*, vol. 211 , no. 1, pp. 531-549, 2013.

[GRA 96] GRABISCH M., "The Application of Fuzzy Integrals in Multicriteria Decision-Making", *European Journal of Operational Research*, no. 89, pp. 445-456, 1996.

[GRA 05] GRABISCH M., LABRENCHE CH., "Fuzzy Measures and Integrals in MCDA", chapter in [FIG 05a], pp. 563-608, 2005.

[GRA 06] GRABISCH M., KOJADINOVIC I., MEYER P., "Using Kappalab R Package for Choquet Integral-based Multiattribute Utility Theory", *International Conference on Processing and Management of Uncertainty (IPMU' 06)*, Paris, France, pp. 1702-1705, 2006.

[GRA 08] GRABISCH M., "L'utilisation de l'intégrale de Choquet en aide multicritère à la décision", *European Working Group "Multiple Criteria Decision Making"*, series 3, no. 14, pp. 5-10, 2008.

[GRE 93] GREFENSTETTE J.J., Chapitre: "Deception Considered Harmful", in *Foundation of Genetic Algorithms 2*, Ed. G. Rawlings & L.D. Whitley, Morgan Kaufmann Editions, USA, 1993.

[GRE 13] GRECO S., KNOWLES J., MIETTINEN K., ZITZLER E. (Eds.), "Learning in Multiobjective Optimization", *Dagstuhl Seminar 12041*, *Dagstuhl Reports*, vol. 2, no. 1, Schloss Dagstuhl, Germany, 2013.

[HAI 96] HAIMES Y., LI D., TULSIANI V., "Multiobjective Decision Tree Method", *Risk Analysis*, vol. 10, no. 1, pp. 111-129, 1996.

[HAM 05] HAMMUMI M., GHÉDIRA K., "The tunneling Algorithm for the K-Graph Partitioning Problem", *Procedings of the 17th IMACS World Congress*, *Paris, France*, 2005.

[HER 00] HERRERA F., HERRERA-VIEDMA E., "Lingusitic Decision Analysis: Steps for Solving Decision Problems under Linguistic Information", *Fuzzy Sets and Systems*, no. 115, pp. 67-82, 2000.

[HOF 95] HOF P. VAN DEN, SCHRAMA R., "Identification and Control", *Automatica, Closed Loop Issues*, December 1995.

[HOL 75] HOLLAND J.H., *Adaptation in Natural and Artificial Systems*, First Edition: University of Michigan Press, U.S.A, 1975.

[HOL 92] HOLLAND J.H., *Adaptation in Natural and Artificial Systems*, Second Edition: MIT Press, Cambridge, Massachusetts, U.S.A, 1992.

[HOW 84] HOWARD R., MATHESON J.R., "Influence Diagrams", chapter in: *Readings on the Principles and Applications of Decision Analysis Strategy*, vol. II (Eds. R.A. Howard, J.E. Matheson), Strategic Decision Group, Menlo Park, CA, U.S.A., 71976, 1984.

[HOW 05] HOWARD R., MATHESON J.R., "Influence Diagrams", *Decision Analysis*, vol. 2, no. 3, pp. 127-143, 2005.

[HWA 81] HWANG C.I., YOAN K. "Multi-Attribute Decision Making: a State of the Art Survey", in: *Lecture Notes in Economics and Mathematics Systems*, Springer Verlag, Berlin-Heidelberg, no. 186, 1981.

[ILG 12] ILGIN M.A., GUPTA S.M., *Remanufacturing Modelling and Analysis*, CRC Press. Taylor and Francis, 2012.

[ISH 13] ISHIZAKA A., NEMERY PH., *Multicriteria Decision Aid: Methods and Software*, John Wiley & Sons, Chichester, U.K., 2013.

[JOH 05] JOHNSON P.E., *Voting Systems*, University of Kansas, 2005.

[JON 75] DE JONG K., *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*, Ph.D. Thesis, University of Michigan, Ann Arbor, USA, 1975.

[KAL 60] KALMAN R.E., "A New Approach to Linear Filtering and Prediction Problems", *Transactions of ASME, Journal of Basic Engineering*, vol. 82D, pp. 35–45, 1960.

[KAL 61a] KALMAN R.E., "Contributions to the theory of optimal control", *Bolletino de la Società Mathematica Mexicana*, no. 5, pp. 102–119, 1961.

[KAL 61b] KALMAN R.E., BUCY R.S., "New Results in Linear Filtering and Prediction Theory", *Transactions of ASME, Journal of Basic Engineering*, Series D, vol. 83, pp. 95-108, 1961.

[KAR 05] KARABOGA D., *An idea based on honey bee swarm for numerical optimization*, Technical Report TROG, Ercizes University, Computer Engineering Department, 2005.

[KEE 92] KEENEY R.L., *Value-Focused Thinking*, Harward University Press, Cambridge, Massachussets, U.S.A., 1992.

[KEE 94] KEENEY R.L., "Creativity in Decision Making with Value Focused Thinking", *Sloan Management Review*, pp. 33-41, Summer,1994.

[KEE 99] KEENEY R.L., RAIFFA H., *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*, Cambridge University Press, U.K., 1999.

[KEL 96] KELMAN A., *Modèles flous pour l'agrégation de données et l'aide à la décision*, Thèse de Doctorat, Université de Paris VI, France, 1996.

[KEN 95] KENNEDY J., EBERHART R.C., "Particle Swarm Optimization", *Proceedings of the IEEE International Conference on Neural Networks, Piscataway, U.S.A.*, vol. IV, pp. 1942-1948, 1995.

[KHA 11] KHAN K., NIKOV A., SAHAI A., "A Fuzzy Bat Clustering Method for Ergonomic Screening of Office Workplaces", *Advances in Intelligent and Soft Computing*, vol. 101, pp. 59-66, 2011.

[KIE 92] KIEFER K., WOLFOVITZ W., "Stochastic Estimation of the Maximum of a Regression Function", Annals of Mathematical Statistics, 1992.

[KIR 83] KIRKPATRICK S., GELATT C., VECCHI M., "Optimization by simulated annealing", *Science*, vol. 220, no. 4598, pp. 671-680, 1983.

[KIR 97] KIRKWOOD C.W., *Strategic Decision Making: Multiobjective Decision Analysis with Spreadsheets*, Duxbury Press, Belmont, U.S.A., 1997.

[KOK 11] KÖKSALAN M., WALLENIUS J., ZIONTS J., *Multiple Criteria Decision Making. From History to the First Century*. World Scientific, 2011.

[KOZ 92] KOZA J.R., *Genetic Programming: on the Programming of Computers by Means of Natural Selection*, MIT Press, U.S.A., 1992.

[KOU 11] KOU G., SHI Y., WANG S., "Multi-criteria Decision Making and Decision Support Systems – Guest Editor's Introduction", *Decision Support Systems*, vol. 51, no. 2, pp. 247-249, 2011.

[KRO 11] KROESE D. P., TAIMRE T., BOTEV Z.I., *Handbook of Monte Carlo Methods*, John Wiley & Sons, New York, U.S.A., 2011.

[KWA 72] KWAKERNAAK H., SIVAN R., *Linear Optimal Control Systems*, Wiley-Interscience, 1972.

[LAN 95] LANDAU I.D., *Identification et Commande des Systèmes*, Hermès, Paris, 1995.

[LEE 72] LEE S.M., *Goal Programming for Decision Analysis*, Auerbach, Philadelphia, 1972.

[LEM 11] LEMMA T.A., "Use of Fuzzy Systems and Bat Algorithm for Energy Modelling in a Gas Turbine Generator", *IEEE Colloquium on Humanities, Science and Engineering, CHUSER'2011*, pp. 305-310, 2011.

[LEV 85] LEVY A.V., GONEY L S., "The tunneling method applied to global optimization", *Numerical Optimization – SIAM Review*, pp. 213-244, 1985.

[LUK 09] LUKASIK S., ZAK S., "Firefly algorithm for continuous constrained optimization tasks", *Lecture notes in computer science*, no. 5796, pp. 97-106, 2009.

[MAL 93] MALLAT S., ZHANG S., "Matching Pursuits with Time-Frequency Dictionaries", *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, December 1993.

[MAN 12] MANSOURI POOR M., SHISHEH SAZ M., "Multi-Objective Optimization of Laminates with Straight Free Edges and Curved Free Edges by Using Bees Algorithm", *American Journal of Advanced Scientific Research*, vol. 1, no. 4, pp. 130-136, 2012.

[MAR 04] MARLER R.T., ARORA J.S., "Survey of Multi-Objective Optimization Methods for Engineering", *Structural and Multi-disciplinary Optimization*. vol. 26, pp. 369-395, 2004.

[MAR 09] MARLER R.T., *A Study of Multi-Objective Optimization Methods for Engineering Applications*. VDM Verlag, Saarbrucken, France, 2009.

[MAR 10a] MARKOVIC Z., "Modification of TOPSIS Method for Solving Multicriteria Tasks", *YUJOR-Yugoslavian Journal of Operations Research*, vol. 10, no. 1, pp. 117-143, 2010.

[MAR 10b] MARLER R.T., ARORA J.S., "The Weighted Sum Method for Multi-Objective Applications; Some Insights", *Structural and Multi-disciplinary Optimization*, vol. 41, no. 6, pp. 853-862, 2010.

[MAY 12] MAYAG B., *The Choquet Integral as a Tool for Aggregating Preferences*. Doctoral Course, U.L.B. Bruxelles, Belgium, 2012.

[MER 87] MERKHOFER M.W., "Quantifying Judgment Uncertainty Methodology, Experiences and Insights", *IEEE Transactions on Systems, Man and Cybernetics*, SMC-17, pp. 741-752, 1987.

[MES 98] MESGHOUNI K., HAMMADI S., BORNE P., "On Modelling Genetic Algorithm for Flexible Job-Shop Scheduling Problems", *Studies in Informatics and Control*, vol. 7, no. 1, pp. 37-47, 1998.

[MES 99] MESGHOUNI K., PESIN P., TRENTESAUX D., HAMMADI S., TAHON C., BORNE P., "Hybrid Approach for Decision Making for Job-Shop Scheduling", *Production Planning and Control*, vol. 10, no. 7, pp. 690-706, 1999.

[MES 02] MESSOC A., MATTSON C.A., "Generating Well-distributed Sets of Pareto Points for Engineering Design Using Physical Programming". *Optimization Engineering*, vol. 3, pp. 431-450, 2002.

[MET 49] METROPOLIS N., ULAM S., "The Monte Carlo Method", *Journal of the American Statistical Association (American Statistical Association)*, vol. 44, no. 247, pp. 335–341, 1949. (DOI:10.2307/2280232. JSTOR 2280232. PMID 18139350.)

[MIL 56] MILLER G., "The Magical Number Seven Plus Minus Two: Some Limits on Our Capacity for Processing Information", *Psychological Review*, no. 63, pp. 81, 1956.

[MIN 80] MINTZBERG H., *The Nature of Managerial Work*, Prentice Hall, Englewood Cliffs, N.J., U.S.A., 1980.

[MIS 12] MISHRA S, SHAW K., MISHRA D., "A new metaheuristic classification approach for microarray data", *Procedia Technology*, vol. 4, pp. 802-806, 2012.

[MIT 95] MITCHELL M., *An Introduction to Genetic Algorithms*, The MIT Press, Cambridge, Massachusetts, U.S.A., 1995.

[MON 10] MONMARCHÉ N., GUINAND F., SIARRY P., *Artificial Ants*, Hardback Press, 2010.

[MUR 91] MUROFUSCHI T., SUGENO M., "A Theory of Fuzzy Measures. Representations of Choquet Integral and Multisets", *Journal of Mathematical Analysis and Applications*, vol. 159, no. 2, pp. 532-549, 1991.

[MUS 13] MUSTAJOKI J., MARTTUNEN M., *Comparison of Multi-criteria Decision Analytical Software*, Report on IMPERA Project, Finish Environmental Institute, Finland, 2013.

[NAS 12] NASIRI B., MEYBODI M.R., "Speciation-based firefly algorithm for optimization in dynamic environments", *International Journal of Artificial Intelligence*, vol. 8, no. 12, pp. 118-132, 2012.

[NEU 53] NEUMANN VON J., MORGENSTERN O., *Theory of Games and Economic Behaviour*, Princetown University Press, Princetown, NJ, 1953

[NIX 91] NIX A.E., VOSE M.D., "Modeling Genetic Algorithms with Markov Chains", *Annals of Mathematics and Artificial Intelligence*, vol. 5, pp. 79–88, 1991.

[OLS 04] OLSON D.L., "Comparison of Weights in TOPSIS", *Mathematical and Computer Modeling*, no. 40, pp. 721-772, 2004.

[ORM 12] OR/MS TODAY, "Decision Analysis Software Survey", 2012. (http://www.orms-today.org/surveys/das/das.html, accessed on 08.01.2014)

[OZB 11] OZBAKIR L., TAPKAN P., "Bee Colony Intelligence in Zone Constrained Two Sided Assembly Line Balancing Problem", *Expert Systems with Applications*, no. 38, pp. 11947-11957, 2011.

[PAR 06] PARETO V., *Manuale di Economica politica*, Societa Editrice Libraria, Milano, *Manual of Political Economy*, A.M. Kelley, New York, U.S.A., 1906 (reprinted in 1971).

[PAR 11] PARPINELLI R.S., LOPES H.S., "New Inspirations in Swarm Intelligence - A Survey", *International Journal on Bio-Inspired Computation*, vol. 3, pp. 1-16, 2011.

[PEA 05] PEARL J., "Influence Diagrams Historical and Personal Perspectives", *Decision Analysis*, vol. 2, no. 4, pp. 232-234, 2005.

[PEN 11] PENG Y., KOU G., SHI Y., "FAMCDM: A Fusion Approach of MCDM Methods to Rank Multiclass Classification Algorithms". *Omega*, no. 39, pp. 677-689, 2011.

[PHA 05] PHAM D.T., GHANBARZADEH A., KOC E., OTRI S., RAHIM S., ZAIDI M., *The Bees Algorithm*, Technical Note, Manufacturing Engineering Centre, Cardiff University, U.K., 2005.

[PHA 09] PHAM D.T., CASTELLANI M., "The Bees Algorithm – Modelling Foraging Behaviour to Solve Continuous Optimisation Problems", Proceedings of Imech E, Part C, vol. 223, no. 12, pp. 2919-2938, 2009.

[PHA 10] PHAM D.T., DARWISH A.H., "Using the Bees Algorithm with Kalman Filtering to Train an Artificial Neural Network for Pattern Classification", *Journal of Systems and Control Engineering*, vol. 224, no. 7, pp. 885 -892, 2010.

[PHA 13] PHAM, D.T., CASTELLANI, M., "Benchmarking and Comparison of Nature-Inspired Population-Based Continuous Optimization Algorithms", *Soft Computing*, pp. 1-33, 2013.

[POL 92] POLYAK B.T., "Acceleration of Stochastic Approximation by Averaging", SIAM Review, vol. 30, 1992.

[POM 99] POMEROL J.CH., BARABA-ROMERO S., *Choix multicritère dans l'entreprise*, Universidad de Alcala, 1996, and Editura Tehnică, Bucharest, Romania, 1999.

[POM 00] POMEROL J-CH., BARABA-ROMERO S., *Multicriterion Decision Making in Management*. Kluwer Academic Publishers, Dordrecht, Germany, 2000.

[POP 01] POPESCU D., SERBANESCU M., "Software Package for Optimal Decisions Large Scale Systems", *IFAC Symposium on Large Scale Systems, LSS 2001*, Bucharest, Romania pp. 225–230, 2001.

[POP 06] POPESCU D., STEFANOIU D. ET AL., *Industrial Automatics*, Romanian Academy Printing House, Bucharest, Romania, 2006 (en roumain).

[RAI 68] RAIFFA H., *Decision Analysis: Introductory Lectures on Choice under Uncertainty*, Addison Wesley, Reading, U.S.A., 1968.

[RAW 91] RAWLINS G., WHITLEY L.D., *Foundations of Genetic Algorithms*, Vol. I, Morgan Kaufmann, U.S.A., 1991.

[RAW 93] RAWLINS G., WHITLEY L.D., *Foundations of Genetic Algorithms*, Vol. II, Morgan Kaufmann, U.S.A., 1993.

[RIA 12] RIABACKE M. DANIELSON M., EKENBERG L., "State-of-Art Prescriptive Criteria Weight Elicitation", *Advances in Decision Sciences*, article ID 276584, DOI:10.1155/2012/276584, 2012.
(http://www.hindawi.com/journals/ads/2012/276584/, accessed on 01.11.2013).

[ROB 71] ROBBINS H., SIEGMUND D., *A Convergence Theorem for Nonnegative Almost Supermartingales and Some Applications*, Optimizing Methods in Statistics, Academic Press, New York, U.S.A., 1971.

[ROY 68] ROY B., "Classement et choix en présence de points de vue multiples (la méthode ELECTRE)", *La Revue d'Informatique et de la Recherche Opérationnelle - RRO*, no. 8, pp. 57-75, 1968.

[ROY 85] ROY B., *Méthodologie multicritère à la décision*, Economica, Paris, France, 1985.

[ROY 91] ROY B., "The Outranking Approach and the Foundations of ELECTRE Method", *Theory and Decisions*, no. 31, pp. 49-73, 1991.
(www.lamsade.dauphine.dauphine.fr/mousseau/prnwiki-2.1.5/uploads/Research/Roy91.pdf, accessed on 25.07.2013)

[ROY 93] ROY B., BOUYSSOU D., *Aide multicritère à la decision. Méthodes et cas.*, Economica, Paris, France, 1993.

[RUS 95] RUSSEL S.J., NORVIG P., *Artificial Intelligence – A Modern Approach*, Prentice Hall, Upper Saddle River, New Jersey, U.S.A, 1995.

[RUZ 13] RUZ G.A., GOLES E., "Learning Gene Regulatory Networks Using the Bees Algorithm", *Neural Computing and Applications*, vol. 22, no. 1, pp. 63-70, 2013.

[SAA 80] SAATY T.L., *The Analytic Hierarchy Process*, McGraw Hill, New York, U.S.A., 1980.

[SAA 97] SAATY T.L., "A Scaling Method for Priorities in Hierarchical Structures", *Journal of Mathematical Psychology*, no. 15, pp. 234-281, 1997.

[SAY 09] SAYADI F., ISMAIL M., MISRAN N., JUMARI K., "Multi-Objective Optimization Using the Bees Algorithm in Time-Varying Channel for MIMO MC-CDMA Systems", *European Journal of Scientific Research*, vol. 33, no. 3, pp. 411-428, 2009.

[SER 99] SERBANESCU M., POPESCU D., "Optimal Decisions for Multimodel Systems", *CSCC'99 Conference*, Athens, Greece, 1999.

[SHA 53] SHAPLEY L.S., "A value for n-person game", chapter in *Contributions to the Theory of Games*, (Annals of Mathematics Studies, no. 28), Princeton University Press, U.S.A., vol. II, pp. 307-317, 1953.

[SIM 55] SIMON H., "A Behavioural Model of Rational Choice", *The Quarterly Journal of Economics*, vol. LXIX, pp. 99-118, February 1955.

[SIM 56] SIMON H., "Rational Choice and the Structure of the Environment", *Psychological Review*, vol. 63, no. 2, pp. 129-138, 1956.

[SIM 57] SIMON H., *Models of Man*, John Wiley & Sons, New York, U.S.A., 1957.

[SIM 60] SIMON H., *The New Science of Management Decisions*, Harper & Row, New York, U.S.A., 1960 (revised edition: Prentice Hall, Englewood Cliffs, N.J., U.S.A., 1977).

[SOD 89] SÖDERSTRÖM T., STOICA P., *System Identification*, Prentice Hall, London, U.K., 1989.

[STA 79] STADLER W., "A Survey of Multicriteria Optimization or the Vector Maximum Problem. Part I, 1776 – 1960", *Journal of Optimization Theory and Applications*, vol. 29, no. 1, pp. 1-52, 1979.

[STA 87] STADLER W., "Initiators of Multi-Criteria Optimization", in: *Recent Advances and Historical Development of Vector Optimization* (J. Jahn, W. Krabs, eds.), *Lecture Notes in Economics and Mathematical Systems*, Springer Verlag, Berlin, no. 294, pp. 3-25, 1987.

[STA 88] STADLER W., "Fundamentals of Multi-Criteria Optimization", in *Multi-Criteria Optimization in Engineering and the Sciences*, Plenum Press, New York, pp. 1-25, 1988.

[STE 03a] STEFANOIU D., IONESCU F., *Vibration Faults Diagnosis by Using Time-Frequency Dictionaries*, Research Report AvH-FHKN-StIo0302, Alexander von Humboldt Foundation & University of Applied Sciences in Konstanz, Germany, February 2003.

[STE 03b] STEFANOIU D., IONESCU F., "Faults Diagnosis through Genetic Matching Pursuit", *The Seventh International Conference on Knowledge-Based Intelligent Information & Engineering Systems KES 2003, Oxford, U.K.*, vol. I, pp. 733-740, September 3–5, 2003.

[STE 05] STEFANOIU D., CULITA J., STOICA P., *A Foundation to System Modeling and Identification*, Éditions Printech, Bucharest, Romania, 2005 (en roumain).

[STE 09] STEFANOIU D., CULITA J., IONESCU F., "Prédire avec des Ondelettes Orthogonales", *Revue Électronique des Sciences et Technologies de l'Automatique (eSTA), Issue spéciale CIFA-2008*, 2-ème partie, vol. 6, no. 2, Papier #8, 2009 (http://www.e-sta.see.asso.fr/?lire=62&sm=6).

[STE 10a] STEFANOIU D., POPESCU D., STANASILA O., *Wavelets – Theory and Applications*, Romanian Academy Press, Bucharest, Romania, 2010 (en roumain).

[STE 10b] STEFANOIU D., CULITA J., "Multi-Variable Prediction of Physical Data", *"Politehnica" University of Bucharest Scientific Bulletin, A Series – Applied Mathematics and Physics*, vol. 72, no. 1, pp. 95-102, 2010.

[STE 13] STEFANOIU D., CULITA J., TUDOR F.S., *Experimental Approaches to Processes and Phenomena Identification*, AGIR Press, Bucharest, Romania, 2013 (en roumain).

[STE 89] STEURER R.E., *Multiple Criteria Optimization: Theory, Computation and Application*. R. E. Krieger Publishing, Malabar, 1989.

[STU 97] STÜTZLE T., HOOS H.H., "Improvements on the Ant System: Introducing Max-Min Ant System", *Proceedings of International Conference on Neural Networks and Genetic Algorithms, Vienna, Austria*, Springer Verlag, 1997.

[STU 00] STÜTZLE T., HOOS H.H, "MAX MIN Ant System", Future Generation Computer Systems, vol. 16, pp. 889–914, 2000.

[SUG 97] SUGENO M., "Fuzzy Members and Fuzzy Integrals – A Survey", chapter in: *Fuzzy Automata and Decision Process* (eds. M.M. Gupta, G.N. Saridis, P. Gaines), pp. 89-102, 1997.

[TAL 09] TALBI E.G., *Metaheuristics: From Design to Implementation*. John Wiley and Sons , Chichester, U.K., 2009.

[TER 91] TERTISCO M., POPESCU D., RUSS I., JORA B., *Automatisations Industrielles Continnues*, Editura Tehnica, Bucharest, Romania, 1991.

[TER 05] TERESKO V., LOENGAROV A., "Collective decision – making in honey bee foraging dynamics", *Computing and Information Systems, University of the West of Scotland, U.K.*, vol. 9, no. 3, pp. 1-7, 2005.

[TOD 02] TODD M.J., "Probabilistic Models for Linear Programming", *Mathematics of Operations Research*, vol. 16, 1991.

[TZE 11] TZENG G.-H., HUANG J.J., *Multiple Decision Making; Methods and Applications*, CRC Press, Taylor and Francis Group, A Chapman & Hall Book, Boca Raton, London, U.S.A.-U.K., 2011.

[TZE 12] TZENG G.-H., LIOU J.J.H., "Comments on "Multiple Criteria Decision Making (MDDM) Methods in Economics: an Overview"", *Technological and Economic Development of Economy*, vol. 18, no. 4, pp. 672-695, 2012.

[TZE 13] TZENG G.-H., HUANG J.J., *Fuzzy Multiple Objective Decision*, CRC Press, Taylor and Francis Corp., Boca Raton, London, U.S.A.-U.K., 2013.

[VIN 81] VINCENT T.L., GRANTHAM W.J., *Optimality in Parametric Systems*, J. Wiley & Sons, New York, 1981.

[VIN 92] VINCKE PH., *Multi-Criteria Decision Aid*. John Wiley & Sons, Chichester, U.K., 1992.

[VOS 91] VOSE M.D., "Generalizing the Notion of Schema in Genetic Algorithms", *Artificial Intelligence*, vol. 50, pp. 385–396, 1991.

[WEN 99] WENZEL W., HAMACHER K., "Stochastic tunneling approach for global minimization of complex potential energy landscape", *Physical Review Letters*, vol. 82, no. 15, pp. 3003-3007, 1999.

[WIS 72] WISMER D.A., *Optimization Methods for Large Scale Systems with Applications*, Mc Graw-Hill Inc., 1972.

[WOL 97] WOLPERT D.H., MACREADY W.G., "No Free Lunch Theorems for Optimization", IEEE Transactions on Evolutionary Computation, vol. 1, no. 1, pp. 67–82, 1997.

[XUW 12] XU W., ZHOU Z., PHAM D.T., LIU Q., JI C., MENG W., "Quality of service in manufacturing networks: a service framework and its implementation", *International Journal Advanced Manufacturing Technology*, vol. 63, no. 9-12, pp. 1227-1237, 2012.

[YAG 88] YAGER R.R., "On ordered weighted averaging aggregation operators in multicriteria decision making", *IEEE Transactions on Systems, Man & Cybernetics*, vol. 18, pp. 183-190, 1988.

[YAN 08] YANG X.S., *Nature inspired metaheuristic algorithms*, Luniver Press, U.K., 2008.

[YAN 09] YANG X.S., "Firefly algorithm for multimodal optimization", *Stochastic algorithms : Foundation and applications, SAGA 2008, Lectures notes in computer science*, no. 5792, pp. 169-178, 2009.

[YAN 10a] YANG X.S., "Firefly Algorithm, Lévy Flights and Global Optimization", in *Nature-Inspired Metaheuristic Algorithms*, Second edition, Luniver Press, pp. 209-218, 2010.

(http://znjs.jpkc.cc/uploads/znjs/file/Firefly%20Algorithm,%20L%C3%A9vy%20Flights%20and%20Global%20Optimization.pdf, accessed on 15.01.2014)

[YAN 10b] YANG X.S., "A New Metaheuristic Bat-Inspired Algorithm", *Nature Inspired Cooperative Strategies for Optimization, NISCO 2010*, in *Studies in Computational Intelligence* (Eds. J.R. Gonzalez et al.), Springer, Berlin, Germany, no. 284, pp. 65-74, 2010.

[YAN 12a] YANG X.S., HOSSEINI S.S., GANDOMI A.H., "Firefly Algorithm for Solving Non-Convex Economic Dispatch Problems with Valve Loading Effect", *Applied Soft Computing*, vol. 12, no. 3, pp. 1180-1186, 2012.

[YAN 12b] YANG X.S., GANDOMI A.H., "Bat Algorithm: a Novel Approach for Global Engineering Optimization", *Engineering Computations*, vol. 29, no. 5, pp. 464-483, 2012.

[ZAR 13] ZARATÉ P., *Tools for Collaborative Decision-Making*, John Wiley and Sons, Hoboken, USA, ISTE, London, UK, 2013.

[ZAV 11] ZAVADSKAS E.K. TURSKIS Z., "Multiple Criteria Decision Making (MCDM). Methods in Economics: an Overview". *Technological and Economic Development of Economy*, vol. 17, no. 2, pp. 397-427, 2011.

[ZAV 14] ZAVADSKAS E.K., "State of art surveys of overviews on MCDM/MADM methods". *Technological and Economic Development of Economy*, vol. 20, no. 1, pp. 165-179, 2014.

[ZID 83] ZIDAROIU C., *Linear Programming*, Editura Tehnica, Bucharest, Romania, 1983 (in Romanian).

[ZIO 88] ZIONTS S., "Multiple Criteria Mathematical Programming: an Updated Overview and Several Approaches", in: *Mathematical Models for Decision Support* (Ed. G. Mitra), NATO ASI Series, vol. F48, Springer Verlag, Berlin, Germany, pp. 135-167, 1988.

[ZRI 08] ZRIBI N., EL KAMEL A., BORNE P., "Minimizing the Makespan for the MPM Job-Shop with Availability Constraints", *International Journal of Production Economics*, vol. 112, no. 1, pp. 151-160, 2008.