Chapter 3

# Metaheuristics and scheduling

## 3.1. Introduction

Scheduling involves taking decisions regarding the allocation of available capacity or resources (equipment, labour and space) to jobs, activities, tasks or customers over time. Scheduling thus results in a time-phased plan, or schedule of activities. The schedule indicates what is to be done, when, by whom and with what equipment. Although the statement of a manufacturing organizational problem such as scheduling may seem simple, we must never underestimate the effort necessary to find its solution [WID 98]. Scheduling problems can be modeled as assignment problems, which represent a large class of combinatorial optimization problems. In most of these cases, finding the optimal solution is very difficult. In fact, except for a few exceptions, the only known method to solve the problem to optimality would be to enumerate an exponential number of possible solutions! Specialists in this case speak of NP-complete problems [CAR 88, GAR 79]. In these conditions, it is necessary to find a solution method providing solutions of good quality in a reasonable amount of time: this is what heuristic methods are all about.

This chapter focuses on describing the three main heuristic classes, *constructive* methods, *local search* methods and *evolutionary* algorithms [COS 95c]. Since these methods are general enough to be applied to a multitude of combinatorial optimization problems, they are called *metaheuristics*.

This chapter is organized as follows: the next two sections indicate how to model scheduling problems in terms of combinatorial optimization and briefly describe the

Chapter written by Marino WIDMER, Alain HERTZ and Daniel COSTA.

major issues encountered when solving such problems. A complete presentation of the main metaheuristics is the subject of section 3.4, preceding a detailed analysis of the application of a tabu search method for the job shop scheduling problem with tooling constraints.

## 3.2. What is a combinatorial optimization problem?

Combinatorial optimization is the field of discrete mathematics involving the resolution of the following problem.

Let X be a set of solutions and $f$ a function that measures the value of each solution in X. The objective is to determine a solution $s^* \in$ X minimizing $f$, i.e.:

$$f(s^*) = \min_{s \in X} f(s)$$

Set X is presumed finite and is in general defined by a set of constraints. As an example, for a job scheduling problem on one machine, X can be made up of all job sequences satisfying precedence and priority constraints while $f$ can correspond to the date at which the last job is finished (makespan).

## 3.3. Solution methods for combinatorial optimization problems

Despite the permanent evolution of computers and the progress of information technology, there will always be a critical size for X above which even a partial listing of feasible solutions becomes prohibitive. Because of these issues, most combinatorial optimization specialists have focused their research on developing heuristic methods. A heuristic method is often defined [NIC 71] as a procedure that uses the structure of the considered problem in the best way possible in order to find a solution of reasonably good quality in as little computing time as possible. General properties of these techniques and the circumstances in which they apply are described in [MÜL 81, SIL 80].

Most difficult combinatorial optimization problems are NP-complete (see Chapter 2). Since no efficient algorithm for solving NP-complete problems is known despite the numerous efforts of these last fifty years, the use of heuristic methods is completely justified when facing such difficult problems. The performance of a heuristic method typically depends on the quality of the solution produced as output and on the computing time necessary to reach such a solution. A compromise has to be found on a case-by-case basis according to the considered optimization problem since these two criteria are of opposite nature. Since the optimal solution value of a large scale difficult combinatorial optimization problem is typically unknown, the quality of a solution produced by an algorithm is typically evaluated using bounds or estimates of this optimal solution value. It might also be interesting to analyze the performance of a given heuristic method in the worst case scenario. A measure of the biggest possible error (with respect to the optimal value) is particularly useful when the goal is to specify in which circumstances the considered heuristic method should not be used.

Although obtaining an optimal solution is not guaranteed, the use of a heuristic method provides multiple advantages when compared with exact methods:

– the search for an optimal solution can be inappropriate in certain practical applications for many reasons such as the large size of the considered problem, the dynamic characterizing the work environment, a lack of precision in data collection, difficulties encountered when formulating the constraints in mathematical terms, the presence of contradictory objectives;

– when applicable, an exact method is often much slower than a heuristic method, which generates additional computing costs and a typically very long response time;

– research principles at the basis of a heuristic method are generally more accessible to inexperienced users. The lack of transparency that characterizes certain exact methods requires the regular intervention from specialists or even from the methods' designers;

– a heuristic method can be easily adapted or combined with other types of methods. This flexibility increases the range of problems to which heuristic methods can be applied.

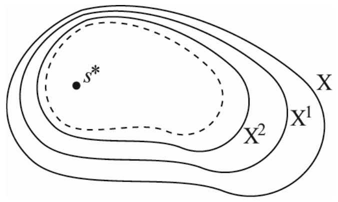
Even though a good knowledge of the problem to be solved is the main contributor to an efficient and successful use of a heuristic method, there are several general rules that can be used to guide the search in promising regions of the solution space X. Guidelines for the use of heuristics in combinatorial optimization can be found in [HER 03]. A classification of heuristic methods was proposed by Zanakis *et al.* [ZAN 89]. In the next sections of this chapter, we describe three fundamentally different heuristic approaches. The research principles of these approaches constitute a basis for several known heuristic methods such as the greedy algorithm, tabu search, simulated annealing and genetic algorithms. The Committee on the Next Decade of Operations Research [CON 88] declared in 1988 that these last three methods were very useful and promising for the solution of a large number of practical applications in the future. This prediction has turned out to be true. Very general in their concept, these methods do, however, require a large modeling effort if we wish to obtain good results.

## 3.4. The different metaheuristic types

### 3.4.1. *The constructive approach*

Constructive methods produce solutions in the form of $s = (x_1, x_2, ..., x_n)$, starting from an initial empty solution $s[0]$ and then inserting at each step $k$ ($k = 1, ..., n$), a component $x_{o(k)}$ ($o(k) \in \{1, 2, ..., n\} \backslash \{o(1), o(2), ..., o(k-1)\}$) in the current partial solution $s[k-1]$. The decision that is taken at a given step is based on the index of the inserted component and on the value to give it. This decision is then never reassessed. The vector representation $s = (x_1, x_2, ..., x_n)$ is quite suitable in solutions for a general assignment problem. Vector positions correspond to objects, whereas each component $x_i$ defines the resource allocated to object $i$.

The exploration of the solution space X with a constructive method is represented in Figure 3.1. The goal is to decrease the size of the problem at each step, which means confining ourselves to an increasingly smaller subset $X^k \subseteq X$. A constructive method finds an optimal solution when each considered subset contains at least one optimal solution $s^* \in X$. Unfortunately, cases where such a condition is always fulfilled are rare. The majority of constructive methods are greedy: at each step, the current solution is completed in the best way possible without taking into consideration all consequences that this generates with regards to final solution cost. In this sense, greedy style methods are often seen as myopic.



$X^k = \{s = (x_1, x_2, ..., x_n) \in X \mid \text{components } x_{o(1)}, x_{o(2)}, ..., x_{o(k)} \text{ are fixed}\}$ $(k = 1, \ldots, n)$

**Figure 3.1.** *Exploration of X with a constructive approach*

Constructive methods are distinguished by their speed and great simplicity. Solutions are very quickly generated for a given problem without having to use highly sophisticated techniques. However, the main drawback with these methods resides unfortunately in the quality of the solutions obtained. The use of the best choice at each step is a strategy with potential catastrophic long term effects. From a theoretical standpoint, obtaining an optimal solution is only ensured for problems accepting a formulation in terms of matroids [GON 85]. It is thus generally wise to implement procedures anticipating side effects and future consequences caused by decisions made during the completion of a partial solution.

We illustrate constructive methods with the algorithm by Nawaz *et al.* [NAW 83] summarized below and developed for the search of a minimal length sequence in a simple flow shop. The scheduling problem in a simple flow shop is a production problem in which *n* jobs must be executed following the same sequence on each one of the *m* machines in the shop; these jobs all have the same process plan but not the same processing times. The processing time of job *i* on machine *j* is denoted $p_{ij}$ ($i = 1 \ldots n, j = 1 \ldots m$).

```
1. for each job i (i = 1 ... n), set $P_i \leftarrow \sum_{j=1}^{m} p_{ij}$
2. sort the jobs in a list in descending order of $P_i$
3. take the first two jobs from the list at step 2 and find the best sequence for these two
   jobs by evaluating both scheduling possibilities. The relative positions of these two
   jobs cannot be modified during the next heuristic phases; set $i \leftarrow 3$;
4. take the job in $i^{th}$ position from the list at step 2 and find the best sequence by
   inserting this job in one of the i possible positions among the jobs already placed;
5. if $i < n$ then set $i \leftarrow i+1$ and go to step 4;
   otherwise STOP: the sequence found is the NEH heuristic solution
```

**Algorithm 3.1.** *NEH heuristic*

Nawaz *et al.*'s algorithm is based on the assumption that a job with long total processing time has priority over a job with a shorter total processing time. The final sequence is found in a constructive way, by inserting a new job at each step and by finding the best possible position for this new job, without modifying the relative positions of the jobs inserted earlier (see Algorithm 3.1).

This heuristic provides a very good compromise between solution quality and computing time [WID 91b].

### 3.4.2. *Local search approach*

Local search methods are iterative algorithms which explore the solution space X by moving step by step from one solution to another. This type of method begins with a solution $s_0 \in$ X arbitrarily chosen or otherwise obtained from a constructive method. The transition from one solution to another is made according to a series of basic modifications which must be defined on a case-by-case basis. The following notations are taken from the [WER 89] reference. We denote $\Delta$ the set of all possible modifications that can be applied to a solution. A solution $s'$ obtained from $s$ by applying a modification $\delta \in \Delta$ is denoted as $s' = s \oplus \delta$. The *neighborhood* N($s$) of a solution $s \in$ X is defined as the set of solutions that can be obtained from $s$ by applying a modification $\delta \in \Delta$. Specifically, we can write: $N(s) = \{s' \in X \mid \exists\, \delta \in \Delta: s' = s \oplus \delta\}$. This type of examination process is interrupted when one or more stopping criteria are met. Figure 3.2 illustrated the general scheme of a local search method. Consecutive transitions from one solution to a neighbor solution define a path in the solution space X.

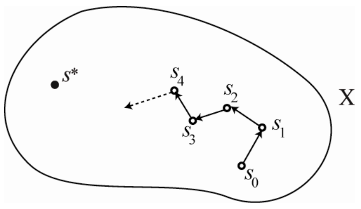


**Figure 3.2.** *Exploration of X with a local search approach*

An oriented graph **G**, called *state space graph*, can be defined to represent a local search. Each solution in X is associated with a vertex in **G**, and an arc from vertex $s_1$ to vertex $s_2$ is introduced in **G** if and only if $s_2$ is a neighbor of $s_1$ (i.e. $s_2 \in N(s_1)$). Modeling an optimization problem and choosing a neighborhood N($s$) must be done in such a way that for each solution $s \in X$, there is at least one path in **G** linking $s$ to an optimal solution $s^*$. The existence of such paths ensures that the local search method can possibly reach an optimal solution starting from any initial solution $s_0$.

The neighborhood of a solution $s$ in the context of assignment problems can be defined in a very simple way [FER 96]. Given a set of objects and a set of resources, and assuming that exactly one resource must be assigned to each object, one can define the neighborhood N($s$) of $s$ as the set of solutions that can be obtained from $s$ by changing the resource assignment of exactly one object.

The descent method described in Algorithm 3.2 is a first example of a local search method. This type of method moves in X by choosing at each step the best neighbor solution in the neighborhood N($s$) of the current solution $s$. This process is repeated as long as the value of the objective function decreases. The search stops when a local minimum is reached. Historically, descent methods have always been among the most popular heuristic methods to handle combinatorial optimization problems. However, they contain two major obstacles which considerably limit their efficiency:

– according to the size and structure of the considered neighborhood N($s$), the search for the best neighbor solution in N($s$) is a problem which can be as difficult as the original problem (which is to find the best solution in X);

– a descent method is unable to escape the first local minimum encountered. Combinatorial optimization problems however typically contain numerous local *optima* with an objective function value which can be very far from the optimal value. This is illustrated in Figure 3.3.

In order to cope with these deficiencies, more sophisticated local search methods have been developed in the last twenty years. These methods accept neighbor solutions with a lesser quality than the current solution in order to avoid local minima of function *f*. The most famous methods of this type are presented in the next sections. The main differences between these methods concern the rule used to choose a neighbor solution and the stopping criteria. The search can be stopped when a solution deemed close enough to being optimal is found. Unfortunately the optimal value of a combinatorial optimization problem is typically not known, and it is therefore often not possible to use such a stopping criterion.

```
Initialization
    choose an initial solution $s \in X$;
    set $s^* \leftarrow s$;

Iterative process
    repeat
        generate $N(s)$;
        determine $s' \in N(s)$ such that $f(s') = \min_{s'' \in N(s)} f(s'')$;
        set $s \leftarrow s'$;
        if $f(s) < f(s^*)$ then set $s^* \leftarrow s$;
    until $s \neq s^*$
```

**Algorithm 3.2.** *The descent method*

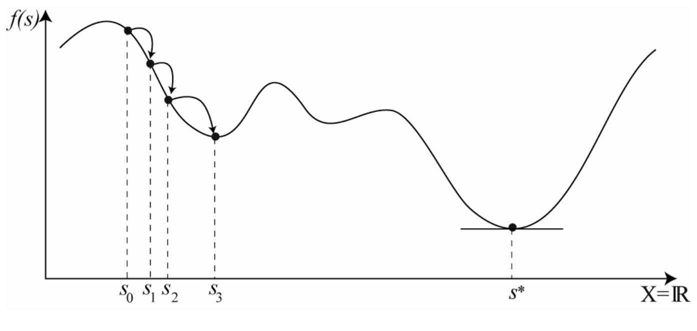


**Figure 3.3.** *A descent method cannot escape a local minimum*

The methods presented below are generally much more powerful than a simple descent method but also much more expensive in terms of computing resources. Their implementation must consider the maximum response time authorized by the program user. We should note in conclusion that significant effort is necessary to correctly adjust parameters used by these methods in order to effectively guide the search throughout X.

### 3.4.2.1. *Simulated annealing*

Originally, the simulated annealing method goes back to experiments by Metropolis *et al.* [MET 53]. Their studies have led to a simple algorithm to simulate the evolution of an unstable physical system toward a state of thermal balance with a fixed temperature *t*. The state of the physical system is characterized by the exact position of all the atoms it contains. Metropolis *et al.* use a Monte Carlo method to generate a series of consecutive system states starting with a given initial state. Any new state is obtained when an atom executes an infinitesimal random movement. Let $\Delta E$ be the energy difference generated by such a disruption. The new state is accepted if the system energy decreases ($\Delta E < 0$). Otherwise ($\Delta E \geq 0$), it is accepted with a certain probability:

$$\mathrm{prob}(\Delta E, t) = \exp\left(\frac{-\Delta E}{k_B \cdot t}\right)$$

where *t* is the system temperature and $k_B$ a physical constant known as Boltzmann constant. At each step, acceptance of a new state where the energy is not lower than the current state is decided by randomly generating a number $q \in [0,1[$. If *q* is lower or equal to prob($\Delta E$,t), then the new state is accepted. Otherwise, the current state is maintained. Metropolis *et al.* have shown that a repeated use of this type of rule brings the system to a state of thermal balance. Many years passed after these studies from Metropolis *et al.* before the simulation algorithm they developed was used to define a new heuristic method for the resolution of a combinatorial optimization problem. Simulated annealing is a local search method where the search mechanism is modeled on the Metropolis *et al.* algorithm and principles of thermodynamic annealing. The idea consists of using the Metropolis *et al.* algorithm with decreasing temperature values *t*. The progressive cooling of a particle system is simulated by on the one hand, making an analogy between the system energy and the objective function of a combinatorial optimization problem, and between system states and the solutions of the considered problem on the other hand. To reach states with as little energy as possible, the system is initially brought to a very high temperature and then slowly cooled down. When the temperature goes down, atom movements become less random and the system will tend to be in low energy states. System cooling must be carried out very slowly in order to reach a state of balance at each temperature *t*. When no new state is accepted at a given temperature *t*, the system is considered as frozen and it is presumed that it has reached a minimum level of energy.

Kirkpatrick *et al*. [KIR 83] and Cerny [CER 85] were the first to follow such a technique to solve combinatorial optimization problems. The neighborhood N($s$) of a solution $s \in$ X contains all states which can be obtained from the current state by slightly moving one atom of the physical system. Only one neighbor solution $s' \in$ N($s$) is generated at each iteration. This one is accepted if it is better than the current solution $s$. Otherwise, one proceeds as with the Metropolis *et al.* algorithm and the new $s'$ solution is accepted with a probability prob($\Delta f$,$t$) depending on the importance of the damage $\Delta f = f(s') - f(s)$ and on a parameter $t$ corresponding to the temperature. Changes in temperature are performed on the basis of a precise cooling pattern. Generally speaking, the temperature is lowered in steps each time a specific number of iterations are executed. The best solution found is memorized in variable $s^*$. The algorithm is interrupted when no neighbor solution has been accepted during a complete constant temperature iteration cycle. The simulated annealing method is described in Algorithm 3.3.

Simulated annealing performance is closely linked to the cooling pattern involved. Numerous theoretical studies have been carried out on this subject and several variations have been proposed [COL 88, OSM 94]. Simulated annealing can be described in the form of a non-homogenous Markov chain [AAR 89, VAN 87], which leads to interesting results in terms of asymptotic convergence of the algorithm when certain specific conditions are met. Geman *et al.* [GEM 84] have demonstrated that the method converges in terms of probability to an optimal solution if temperature $t_k$ at the $k^{th}$ iteration meets the two following conditions:

i) $\lim\limits_{k \to \infty} t_k = 0$

ii) $t_k \geq \dfrac{c}{\log(k+1)} \qquad \forall\, k = 1, 2, \ldots$

where c is a constant independent of k. Hajek [HAJ 88] established a similar convergence result by asserting:

iii) $t_k = \dfrac{c}{\log(k+1)} \qquad \text{and} \qquad c = \max\limits_{s \in X} f(s) - \min\limits_{s \in O} f(s)$

where O represents the set of all local minima of $f$ which are not optimal solutions. In general, constant $c$ is called the maximum depth of function $f$.

In reality, the above condition is difficult to obtain and very costly in terms of computation. We generally prefer the cooling pattern by levels shown in Algorithm 3.3, even though it does not guarantee algorithm convergence to an optimal solution.

Lately, the simulated annealing algorithm has been abundantly used to solve practical optimization problems. A detailed review of the literature was carried out by Collins *et al.* [COL 88]. Several simulated annealing algorithm applications are proposed in a book edited by Vidal [VID 93]. The interested reader can refer to references [EGL 90, REE 93, RUT 89, VAN 87] for more detailed information on the simulated annealing.

```
Initialization
    choose an initial solution $s \in X$;
    set $s^* \leftarrow s$;
    set $k \leftarrow 0$;                 (global iteration counter)
    set new_cycle ← true;       (Boolean variable indicating if it is worth
                                executing a new cycle of iterations)
    set $t \leftarrow t_0$;                (t_0 = initial system temperature)
Iterative process
    while new_cycle = true do
    set nbiter ← 0;             (iteration counter internal to a cycle)
    set new_cycle ← false;
    while (nbiter < nbiter_cycle) do
        set $k \leftarrow k + 1$ and nbiter ← nbiter + 1;
        randomly generate a solution $s' \in N(s)$;
        set $\Delta f \leftarrow f(s') - f(s)$;
        if $\Delta f < 0$ then set $s \leftarrow s'$ and new_cycle ← true;
        otherwise
            set $prob(\Delta f, t) \leftarrow exp(-\Delta f / t)$;
            generate a random real number q from an uniform distribution over the
            interval [0,1[;
            if $q < prob(\Delta f, t)$ then set $s \leftarrow s'$ and new_cycle ← true;
        if $f(s) < f(s^*)$ then set $s^* \leftarrow s$;
    set $t := a \cdot t$                   ($0 < a < 1$: cooling factor)
```

**Algorithm 3.3.** *Simulated annealing*

3.4.2.2. *Threshold accepting methods*

Threshold accepting methods are local search methods that can be seen as deterministic variants of simulated annealing. The main difference between these two methods is the level of acceptance of a lower quality solution at each step. In a threshold accepting method, this type of decision is taken in a deterministic manner, without having to use the principles of thermodynamic annealing. It is strictly based on an auxiliary function $\gamma(s,s')$ and on a threshold S which can possibly involve the value $f(s^*)$ of the best solution encountered so far. Function $\gamma(s,s')$ and threshold S can be defined in many ways leading to several variations for threshold accepting methods. The threshold accepting method is presented in a generic way in Algorithm 3.4, followed by three adaptations of this method developed by Dueck and Scheuer [DUE 90, DUE 93]. The algorithm is generally interrupted when a number *nbiter_tot* of iterations is reached or when the best found solution $s^*$ is not improved during a number *nbiter_max* of iterations.

*Standard threshold accepting method*

The threshold S is defined the same way as temperature $t$ in the simulated annealing algorithm. It is initially set at a high value, then proportionally lowered (i.e., S is set equal to $a \cdot S$ with $0 < a < 1$) each time a predetermined number of iterations is completed. The aim of this method is to accept all neighbors $s'$ which do not deteriorate the quality of the current solution $s$ in a significant way. For this purpose, the auxiliary function $\gamma(s,s')$ is defined as $\gamma(s,s') = f(s') - f(s)$.

*Great deluge method*

In this case, threshold S corresponds to a level of water In a minimization problem, a neighbor solution $s'$ is accepted if and only if $f(s')$ is below the water level S regardless of the value $f(s)$ of the current solution. In this way, $\gamma(s,s') = f(s')$, The water level is initially set at a high value $S_0$, and it then decreases linearly (i.e., S is set equal to $S - d$ where $d$ is a parameter measuring the decreasing speed).

```
Initialization
    choose an initial solution $s \in X$;
    set $s^* \leftarrow s$;
    set $nbiter \leftarrow 0$;        (iteration counter)
    set $best\_iter \leftarrow 0$;    (iteration leading to the best $s^*$ solution found to date)
    set $S \leftarrow S_0$;           ($S_0$ = initial threshold)
Iterative process
    while no stopping criterion is met do
        set $nbiter \leftarrow nbiter + 1$;
        randomly generate a solution $s' \in N(s)$;
        if $\gamma(s,s') < S$ then
            set $s \leftarrow s'$;
            if $f(s) < f(s^*)$ then set $s^* \leftarrow s$ and $best\_iter \leftarrow nbiter$;
        update threshold S
```

**Algorithm 3.4.** *Threshold accepting method*

Originally, this method was designed to solve maximization problems. First, the level of water is arbitrarily set at a low value, and it is then linearly increased. In such a case, a neighbor solution *s*' is only retained if it is over the water level (i.e. if *f*(*s'*) > S). This explains the name of the method. Such a principle of research can be simply illustrated by considering a hiker unable to swim and wanting to reach the highest point of a given region when the level of water constantly increases. The hiker's chances for success are higher if the water increase is slow (i.e., *d* is small).

*Record-to-record travel method*

In this adaptation of the threshold accepting method, all neighbor solutions *s'* of significantly lower quality than the best solution *s** found so far are rejected. In order to do this, we must define a maximal deterioration *max_d*, in relation to value *f*(*s**), over which any neighbor solution *s'* is automatically rejected. As in the previous method, acceptance of a new solution is made without examining the value of the current solution, hence $\gamma(s,s') = f(s')$. Threshold S is initialized at a high value and is then updated at each step by setting S equal to $f(s^*) + max\_d$. Parameter *max_d* is typically not modified during the execution of the algorithm.

According to their authors [DUE 90, DUE 93], the three previous methods provide numerous advantages compared to the simulated annealing method. Several tests were carried out in the context of the traveling salesman problem. It would seem that a method of threshold accepting generally gives better results than simulated annealing while being quicker and less sensitive to the set of parameters used. Sinclair [SIN 93] confirmed these observations after using the great deluge and the record-to-record travel methods for balancing hydraulic turbine runners. We do not know of any more comparative studies leading to similar conclusions.

### 3.4.2.3. *Tabu search*

Tabu search is a general iterative method for combinatorial optimization introduced by Glover [GLO 86] in a specific context and later developed in a more general context [GLO 89, GLO 90]. Independently, Hansen developed the SAMD method (for Steepest Ascent Mildest Descent) based on similar ideas [HAN 86]. The tabu search method is very powerful over a considerable number of combinatorial optimization problems, scheduling in particular.

*Method description*

As previously mentioned, the movement from a current solution *s* to a neighbor solution *s'* is chosen in such a way that:

$$f(s') = \min_{s'' \in N(s)} f(s'')$$

As long as we are not in a local *optimum*, any iterative local search method behaves as the descent method and improves the objective function value at each step. On the other hand, when we reach a local *optimum*, the movement rule described earlier makes it possible to choose the lesser of bad neighbors, i.e. the one inducing as little increase of the objective function as possible. The disadvantage that would represent a method only based on this principle is that if a local minimum $s$ is found at the bottom of a deep valley, it would be impossible to get out of it in a single iteration, and a movement from $s$ to a neighbor solution $s' \in N(s)$ with $f(s') > f(s)$ can cause the opposite movement at the next iteration, since generally $s \in N(s')$ and $f(s) < f(s')$; there is therefore a risk of "cycling" around this local minimum $s$. For this reason, the tabu search algorithm uses a second principle consisting of keeping in memory the last visited solutions, and prohibiting a return to them for a fixed number of iterations, the goal being to offer enough time to the algorithm to exit a local minimum. In other words, the tabu search method keeps a list T of "tabu" solutions at each step, and it is prohibited to move to any solution in T. When moving from $s$ to a neighbor solution, the oldest element in T is removed and replaced by $s$. The necessary space to register a list of tabu solutions can be quite large in terms of memory. For this reason, it is sometimes preferable to prohibit a set of movements that would bring back to solutions already visited. These prohibited movements are called *tabu movements*.

*The aspiration function*

During the choice for the best solution $s' \in N(s)$, we may have to decide between several candidates providing the same value to the objective function. If the chosen neighbor does not lead to a good region of the solution space X, it might then be desirable to be able to go back to a visited solution $s$ despite the fact that it is now part of the tabu list T, in order to explore a new region neighboring $s$.

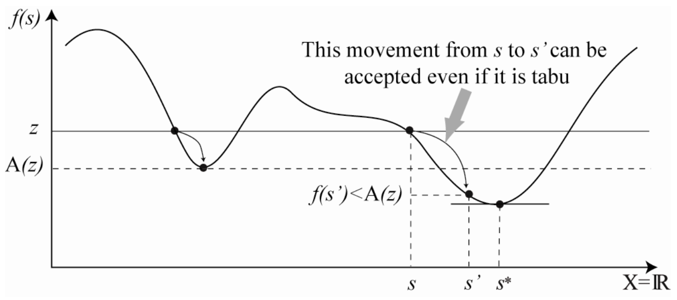

f(s)
This movement from s to s' can be accepted even if it is tabu
z
A(z)
f(s')<A(z)
s
s'
s*
X=ℝ

**Figure 3.4.** *Illustration of the aspiration function concept*

For this reason, the tabu search method involves a new ingredient called *aspiration function* defined over all values of the objective function: when a neighbor solution $s' \in N(s)$ belongs to T and meets the aspiration criterion (i.e. $f(s') < A(f(s))$), we cancel its tabu status and $s'$ then becomes a candidate during the selection for a best neighbor of $s$. In general, $A(f(s))$ takes on the value of the best solution $s^*$ encountered so far (i.e., we "aspire" to determine a better solution than $s^*$). Another possibility is to set $A(z)$ equal to the best value obtained by moving from a solution $s$ with value $f(s)=z$ (see Figure 3.4).

*Neighborhood*

For certain problems, the neighborhood $N(s)$ of a solution s is large and in addition, the only way to determine a solution $s'$ minimizing $f$ over $N(s)$ is to review all solutions in $N(s)$; we thus prefer to generate a subset $N' \subseteq N(s)$ only containing a sample of solutions neighboring $s$ and we choose a solution $s' \in N'$ of minimal value $f(s')$.

*Ending condition*

We must still define a stopping condition. We are generally given a maximum number *nbmax* of iterations between two improvements of the best solution $s^*$ encountered so far. In certain cases, it is possible to determine a lower bound $\underline{f}$ on the optimal value and we can then stop the search when we have found a solution $s$ of value $f(s)$ close to $\underline{f}$.

The tabu search method is summarized in Algorithm 3.5.

```
Initialization
    Choose an initial solution $s \in X$;
    set $s^* \leftarrow s$;
    set $nbiter \leftarrow 0$;      (iteration counters)
    set $T \leftarrow \emptyset$;          (the tabu list is initially empty)
    initialize the aspiration function A;
    set $best\_iter \leftarrow 0$; (iteration where $s^*$ was found)
Iterative process
    while $(f(s) > \underline{f})$ and $(nbiter - best\_iter < nbmax)$ do
    set $nbiter \leftarrow nbiter+1$;
    generate a subset $N' \subseteq N(s)$ of solutions neighboring s;
    choose the best solution $s' \in N'$ solution such that $f(s') \leq A(f(s))$ or $s' \notin T$;
    update the aspiration function A and the tabu list T;
    set $s \leftarrow s'$;
    if $f(s) < f(s^*)$ then set $s^* \leftarrow s$ and $best\_iter \leftarrow nbiter$
```

**Algorithm 3.5.** *Tabu search*

In summary, the following elements are the main ingredients of a tabu search method:

| | |
|---|---|
| X | the set of solutions |
| $f$ | the objective function defined on X |
| N($s$) | the neighborhood of a solution $s \in X$ |
| $\|T\|$ | the number of tabu solutions or the size of the *tabu list* T |
| A | the aspiration function |
| *nbmax* | the maximum number of iterations between two improvements of $s^*$ |
| N' | the subset of N($s$) (the way of generating it and its size) |
| $\underline{f}$ | a lower bound on the objective value. |

Most tabu method ingredients are illustrated in section 3.5. Those not present have been considered as useless during the adaptation of the tabu method to the considered scheduling problems. The reader interested in more illustrations of all ingredients can refer to scientific papers from Glover [GLO 89; GLO 90; GLO 97]. More refined versions of the tabu search algorithm have been presented in more detail in the literature. The interested reader should consult references [GLO 89, GLO 90, GLO 93b, HER 97, REE 93, SOR 94, TAI 93, WER 89] for more information on this subject. Several strategies were proposed to improve the efficiency of the tabu search algorithm presented above [GLO 97]. *Intensification* and *diversification* are two of these strategies.

Intensification consists in a detailed exploration of a region of X deemed promising. Its implementation usually resides in a temporary widening of the neighborhood of the current solution in order to visit a set of solutions sharing common properties. Another possibility consists of returning to the best solution s* encountered so far and to restart the search from this solution with a smaller tabu list size for a limited number of iterations. Diversification is a complementary technique to intensification. Its objective is to direct the search procedure to unexplored regions in X. The simplest diversification strategy is to restart the search process periodically from a solution either randomly generated or judiciously chosen in a region that has not yet been visited. In this way, we decrease the influence of the choice of the initial solution $s_0$ over the global algorithm performance. The same type of effect can be obtained by temporarily modifying the objective function or by favouring modifications not made during a large number of iterations.

To conclude this presentation of the tabu search method, we should mention that no theoretical result guarantees the convergence of the algorithm toward an optimal solution of the considered problem contrary to the simulated annealing method. The main reason for this fact comes from the nature of the method itself. Since the method is highly adaptive and flexible, its analysis with traditional mathematical tools is difficult. The only known theoretical results to this day involve a probabilistic version of the tabu search algorithm close to the simulated annealing method. Faigle and Kern [FAI 92] have shown that it is possible to modify the objective function and the generation probability of neighbor solutions in order to ensure global convergence of the search process. Such a result is unfortunately not very useful in practice because the probability of convergence to an optimal solution only happens after an infinite time period. In addition, the tabu search method doesn't need to converge toward an optimal solution to be effective. The goal is to visit at least one optimal solution or a good quality solution during the search.

Despite the fact that no substantial theoretical result was established, the tabu search algorithm has been arousing increased interest since its discovery twenty years ago. Impressive results were obtained for numerous combinatorial optimization problems. Several examples of tabu search applications are proposed in [GLO 93a]. A detailed example of a tabu search application is the subject of section 3.5: job shop scheduling with tooling constraints. For the conclusion of this section, we should note that Glass and Potts have achieved an interesting comparison of different local search methods for the single flow shop problem [GLA 96], while the same kind of analysis was performed by Galinier and Hertz for the graph coloring problem [GAL 06].

### 3.4.3. *The evolutionary approach*

Life sciences and natural processes have always fascinated engineers. They use structures and mechanisms from reality to develop artificial objects used in various contexts. In the field of combinatorial optimization, natural phenomena complexity has served as a model for increasingly sophisticated algorithms in the last thirty five years. The evolutionary methods presented in this section constitute the basis of a constantly growing field of computer programming.

Contrary to constructive and local search methods involving a single solution (partial or not), evolutionary methods handle a group of solutions at each search process step. The main idea is to regularly use the collective properties of a group of solutions, called population, with the goal of efficiently guiding the search to appropriate solutions in the solution space X. In general, the size $p$ of a population remains constant throughout the process. After the random or constructive method generation of an initial population containing solutions $s_{0,i} \in X$ ($i = 1, ..., p$), an evolutionary method attempts to improve the average quality of the current population by using natural evolution principles. In our terminology, the cyclic process at the basis of an evolutionary method is made up of the continuous succession of a cooperation phase and an individual adaptation phase. This new formalism applies to most evolutionary methods developed to date.

In the cooperation phase, the solutions in the current population are compared and then combined together in order to produce new and good quality solutions for the long term. The resulting information exchange leads to the creation of new solutions which have the predominant characteristics contained in current population solutions. In the individual adaptation phase, the solutions evolve independently respecting a series of predefined rules. Modifications experienced by each one are performed without interaction with other solutions in the population. A new population of solutions is created at the end of each individual adaptation phase.

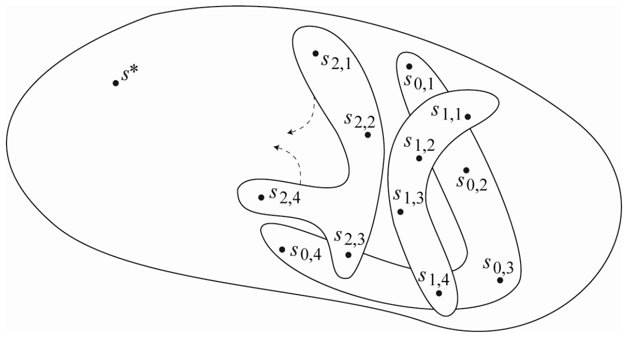


**Figure 3.5.** *Exploration of X with an evolutionary approach*

The search mechanism at the basis of an evolutionary method is briefly represented in Figure 3.5. The goal is to locate the best possible solutions by manipulating at each step a set of solutions located in different promising regions of the solution space X.

In what follows, we say that an evolutionary method prematurely converges or goes through a diversity crisis when the current population contains a high percentage of identical solutions. Different mechanisms can be incorporated to avoid this drawback. The simplest way to prevent premature convergence risks is to introduce a measure $E \in [0, 1]$ based on the notion of entropy [FLE 94, GRE 87] in order to evaluate the degree of diversity within a population of solutions. A value $E = 0$ indicates that the population is made up of identical individuals whereas a value closer to 1 suggests a large diversity in the population. When entropy E is judged to be too low at a given step, the idea is to take measures to reintroduce sufficient diversity the current population.

### 3.4.3.1. *Genetic algorithms*

Genetic algorithms are evolutionary methods greatly influenced by biological mechanisms linked to the principles of natural evolution and selection. Initially developed by Holland *et al.* [HOL 75] to respond to specific needs in biology, genetic algorithms have quickly been adapted to a large variety of contexts. In a simple genetic algorithm [GOL 87], the search is handled by three operators applied consecutively. The cooperation phase is managed by a reproduction operator and a crossover operator whereas the individual adaptation phase uses a mutation operator. It is important to note that concepts at the basis of genetic algorithms are extremely simple. In fact, they only involve randomly generated numbers and a set of generic probabilistic rules which do not necessarily consider all characteristics of the problem discussed. Chapter 4 focuses on genetic algorithms.

#### 3.4.3.2. *Scatter search method*

The scatter search method [GLO 94, GLO 95] simulates an evolution which is not in direct relation with genetics. No restriction is made in how to code solutions. The cooperation phase does not refer to a reproduction operator, and the combination of solutions is more generic than with a genetic algorithm. The combination operator generates new individuals by considering groups of solutions with possibly more than two elements. The role of the individual adaptation phase is to repair individuals produced by the combination operator if they do not satisfy all constraints of the considered problem; during the individual adaptation phase, each individual is also possibly improved by means of an improvement operator. This type of mechanism can be implemented in the form of a local search method involving an auxiliary function penalizing individuals that do not satisfy all constraints of the considered problem. The few variations of scatter search proposed in literature were all perceived to be a generalization of a genetic algorithm [RAD 94]. Scatter search was originally developed by Glover [GLO 77] for the solution of integer programming problems. The idea is to continuously operate linear combinations over a set of solutions coded as vectors with integer components. An auxiliary procedure is applied at each step to repair individuals if the linear combination has produced non integer components in vectors. This repairing process can be done by using systematic rounding or even better, a local search method meant to find an integer solution that is the "closest" to the considered non integer vector. Among successful applications of the scatter search method, we should mention the adaptations made for quadratic assignment problems [CUN 97], neural network training [KEL 96] or unconstrained continuous optimization [FLE 96a].

#### 3.4.3.3. *The ant algorithm*

Collective performance [THE 94] of social insects such as ants, bees, wasps or termites, have intrigued entomologists for a very long time. The main question involves mechanisms enabling individuals of a single colony to manage their activities and to favor survival of the species. Everything is done as if an invisible agent is coordinating activities of all individuals from the center of the colony. Studies have shown that this global behavior was the result of a multitude of particularly simple local interactions. The nature of these interactions, information processing mechanisms and the difference between solitary and social behaviors have long remained a mystery. During the accomplishment of a specific task by an insect colony, it was observed that task coordination did not depend directly on workers but on the state of the task's progress. The worker does not manage its job; it is guided by it. Any working insect modifies the form of stimulation generating its behavior and causes the creation of a new stimulation that will trigger other reactions from it or a coworker.

To illustrate the appearance of collective structures in a society of insects, we must mention the example of an ant colony searching for a food source. Initially, ants leave their nest and move randomly. When an ant happens to discover a food source, it informs its coworkers of its discovery by setting off a temporary mark on the ground upon its return to the nest. This mark is nothing more than a chemical substance called a pheromone, which will guide the other ants toward the same food source. Upon their return, these ants also set off pheromones on the ground and thus reinforce the trail marking leading from the nest to the source of discovered food. Pheromone marking reinforcement in the most used trail optimizes food gathering. In the long run, ants will only use the closest source because the trail leading to remote sources will evaporate and become undetectable. This example shows that the ant colony converges to an optimal solution where each ant only has access to local information and it is unable to resolve the problem on its own in a reasonable amount of time.

In the last few years, engineers have focused on social insect behavior in order to create a new form of "collective problem solution". The ant algorithm, initially developed by Colorni *et al.* [COL 91, COL 92, DOR 96], is an evolutionary method where search mechanisms are greatly influenced by the collective behavior of an ant colony. In the cooperation phase, each current population solution is examined in turn in order to update a global memory. Then, the individual adaptation phase involves a constructive method which uses information contained in the global memory to create new solutions. This type of approach repeatedly uses a constructive method by including the accumulated experience from previous method applications. Each constructive method application corresponds to the work accomplished by a single ant. In the previous example, the global memory appears in the traces of pheromone set off by the ants. This search principle easily adapts to the general assignment problem. Concepts presented below generalize the ideas of Colorni *et al*. [COL 91, COL 92, DOR 96] in the context of the traveling salesman. Consider a set of $n$ objects to which a resource $j \in \{1, ..., m\}$ has to be assigned. This assignment must meet a series of given constraints and be as inexpensive as possible. Two decisions are made at each step of a constructive method. The first one involves the choice of one of the $n$ objects and the second defines the resource assigned to it. In a traditional constructive method, these decisions are made in a manner that will complete the current solution in the best way possible. In other words, they are based on an object and a resource with the maximum appeal for the current step. The ant algorithm uses a complementary notion to an object or resource appeal. Partial solutions are completed in a probabilistic manner by relying on past experience. The term trace will then be used to represent an element of information which is known based on experiments carried out in the completion of previous solutions. At each step $k$ of the development process, an ant chooses an object $i$, not yet assigned, with probability $p_{\text{obj}}(k, i)$ and a resource $j$ that is feasible for object $i$, with probability $p_{\text{res}}(k, i, j)$. Each of these probabilities involves two numbers $\tau(s[k-1],.)$ and $\eta(s[k-1],.)$ which measure the trace and appeal respectively for an object or resource, given a partial solution $s[k-1]$ in which objects $o(1)$, ..., $o(k-1)$ have already been assigned to resources $x_{o(1)}, ..., x_{o(\text{k-1})}$:

$$p_{\text{obj}}(k,i) = \frac{\tau_1(s[k-1],i)\,.\,\eta_1(s[k-1],i)}{\sum\limits_{q \notin \{\text{o}(1),\text{o}(2),\,...,\text{o}(k-1)\}} \tau_1(s[k-1],q)\,.\,\eta_1(s[k-1],q)}$$

$$p_{\text{res}}(k,i,j) = \frac{\tau_2(s[k-1],i,j)\,.\,\eta_2(s[k-1],i,j)}{\sum\limits_{r \text{ is admissible for } i} \tau_2(s[k-1],i,r)\,.\,\eta_2(s[k-1],i,r)}$$

Algorithm 3.6 presents an ant algorithm for the solution of general assignment problems. The number of ants is $n$. During the first cycle all traces are set to 1 because the ants do not know the problem to solve. At the end of a cycle, traces $\tau_1$ and $\tau_2$ are updates in relation to the characteristics of a solution produced by each ant. Generally, only one of the two decisions is made in a probabilistic way at each step of the constructive method. In most applications solved with the help of the ant algorithm, choosing the next object is done either naturally, or based on a deterministic rule not involving the concept of trace.

A simplified version of the ant algorithm was proposed by Colorni *et al.* [COL 94] for the job shop scheduling problem. They construct a graph $G_{\text{jobshop}}$ based on the method of Roy and Sussmann [ROY 64]: each operation to be performed on the machines is a vertex of $G_{\text{jobshop}}$; an artificial vertex (corresponding to the start of the schedule) is added to $G_{\text{jobshop}}$, and linked by an arc to each job's first operation; for each job, two consecutive operations in the process plan are linked by an arc; two operations which can be executed on a single machine are connected by an *edge* (i.e., a non oriented arc). Then for each ant, they apply the procedure described in Algorithm 3.7.

At the end of this procedure, the ant has visited all vertices of the graph (i.e, all job operations) according to the order in which set B was constructed. This sequence makes it possible to define a direction on all edges of the graph, thus defining an ordering of the operations on each machine. Colorni *et al*. then apply the longest route with minimal length algorithm to find the date at which the last job ends [ROY 64]. In addition to its applications to the traveling salesman and the plant problem, the ant algorithm was also adapted to graph coloring [COS 97] and quadratic assignment [MAN 99], among others. More details on ant colony algorithms can be found in [DOR 04].

```
Initialization
$f(s^*) \leftarrow$ infinite;              (infinite = arbitrarily large value)
$ncycles \leftarrow 0$;                    (cycle counter)
$best\_cycle \leftarrow 0$;                (cycle leading to the best $s^*$ solution found to date)
$\tau_1(..., i) := 1$; $\tau_2(..., i, j) := 1$;  $\forall i = 1, ..., n$  $\forall j = 1, ..., m$

Iterative process
while no stopping criterion is met do
    $ncycles \leftarrow ncycles + 1$;
    for $q = 1$ to $nants$ do
        for $k = 1$ to $n$ do
            choose an object $i \in \{1, ..., n\} \setminus \{o(1), ..., o(k-1)\}$ with probability $p_{obj}(k, i)$;
            choose a feasible resource $j$ for object $i$ with probability $p_{res}(k, i, j)$;
            assign resource $j$ to object $i$ ; $x_i := j$; $o(k) := i$;
        compute the cost $f(s_q)$ of solution $s_q = (x_1, ..., x_n)$;
    set population $p$ equal to the set $\{ s_1, s_2, ..., s_{nants} \}$
    set $s' = \arg\min \{f(s) \mid s \in p\}$;
    if $f(s') < f(s^*)$ then $s^* \leftarrow s'$ and $best\_cycle \leftarrow ncycles$;
    update traces $\tau_1$ and $\tau_2$
```

**Algorithm 3.6.** *An ant algorithm for a general assignment problem*

```
Progression of an ant
    Construct the graph $G_{jobshop}$ where $\alpha$ represents the artificial vertex;
    for every vertex $x$ in $G_{jobshop}$ set $A(x)$ equal to the set $\{t \mid$ there is an arc from $x$ to $t$ or
    an edge between $x$ and $t\}$;
    set $A \leftarrow A(\alpha)$ and $B \leftarrow \varnothing$;
    set $C$ equal to the vertex set of $G_{jobshop}$;

Iterative process
    while $C \neq \varnothing$ do
        choose a vertex $t \in A$ according to a probability of transition;
        set $A \leftarrow A \cup A(t) \setminus \{t\}$, $C \leftarrow C \setminus \{t\}$ and $B \leftarrow B \cup \{t\}$;
```

**Algorithm 3.7.** *Progression of an ant*

### 3.4.4. *The hybrid approach*

Evolutionary methods, and particularly genetic algorithms have largely been studied since their very first discoveries early in the 1970s [HOL 75]. The many adaptations proposed in literature fulfill the main weaknesses of traditional evolutionary methods where global performance is often far lower than that of a local search method such as tabu search or simulated annealing. In a more specific context, it is now understood that a simple genetic algorithm cannot provide good results when the set of solutions must satisfy many constraints. Grefenstette [GRE 87] has shown that it is possible to take advantage of the characteristics of the problem studied when defining all operators in a genetic algorithm.

Most of the innovations introduced in the evolutionary method field use concepts no longer related to natural evolution principles. Very interesting results were recently obtained by inserting a local search method into the individual adaptation phase of an evolutionary method. In what follows, we will refer to this new combined search method in terms of hybrid algorithm. The strength of a hybrid algorithm resides in the combination of these two fundamentally different search principles. The role of the local search method is to explore in depth a given region of the set X of solutions whereas the evolutionary method introduces general conduct rules in order to guide the search through X. In this sense, the combination operator has a long term beneficial diversifying effect. To our knowledge, works by Glover [GLO 77], Grefenstette [GRE 87] and Mühlenbein *et al.* [MÜH 88] are the originators of hybrid algorithms. Each one has introduced a simple descent method (see Algorithm 3.2) to increase the performance of an existing evolutionary method. Glover used a scatter search method and Grefenstette and Mühlenbein *et al*. have used a genetic algorithm to solve integer programming and traveling salesman problems respectively. Simulated Annealing and tabu search are improvements of the simple descent method. It is thus normal to use them within an evolutionary method to further increase its performance. Combination possibilities are numerous at this level [CAL 99, COS 95a, COS 95b, FAL 96, FLE 94, FLE 96b, MOS 93, REE 96].

In his thesis, Costa proposes two hybrid algorithms based on the two combination schemes described in Figure 3.6 [COS 95c]. The resulting algorithm is called evolutionary tabu search or evolutionary descent algorithm depending on the local search method used. In the first case, the local search method plays the role of mutation operator whereas in the second case it replaces the reproduction operator. This operator has been deleted to decrease premature convergence risks which are often high in a hybrid algorithm where the local search method is deterministic.

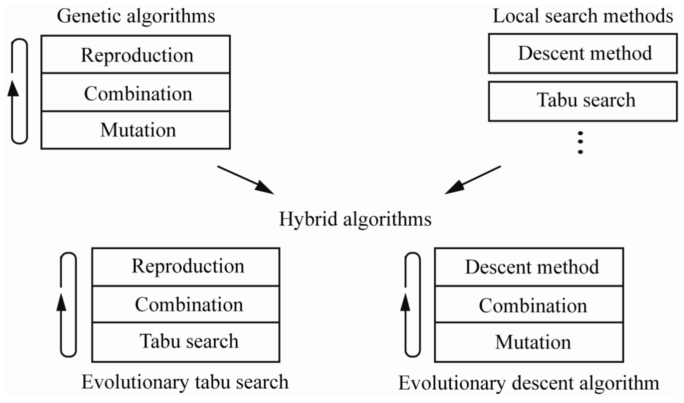


**Figure 3.6.** *Two hybrid schemes using a genetic algorithm*

The evolutionary tabu search has been applied to sports scheduling problems. Costa has shown that the performance of the algorithm is significantly better than the tabu search executed separately during a comparable time period. This type of behavior is the result of the complementarity which exists between tabu search and genetic algorithms. To be really efficient, tabu search requires important modeling and parameter tuning efforts. Conversely, even though genetic algorithms are less efficient, they have the advantage of being robust and based on extremely simple rules. In addition, Costa presents an adaptation of the evolutionary descent algorithm to solve graph coloring problems [COS 95b]. The performance of the algorithm is much better than the genetic algorithm and the descent method used in it. Results obtained from a hybrid algorithm are usually very high in quality. Unfortunately, necessary computing times to reach a given quality solution can become prohibitive. After a comparison of several approaches in the solution of a range of quadratic assignment problems, Taillard [TAI 95] concludes that hybrid algorithms are among the most powerful but also the most time expensive. Choosing the right method largely depends on available time for the solution of a specific problem. Nowadays research focuses on parallelizing hybrid algorithms in order to reduce computing time and solve larger problems.

## 3.5. An application example: job shop scheduling with tooling constraints

Production systems are constantly evolving, especially in the engineering industry. Rigid transfer lines are currently being replaced by cells or flexible workshops. These systems are made up of numerical control machines and can simultaneously produce different types of products. This versatility is essential in the current manufacturing context. In fact, it is important to be able to produce different types of products quickly in order to satisfy client demand. This means that set up times between two types of products must be very short. To reach this goal, optimal management of tool switches is necessary. Because of this, one basic hypothesis of the scheduling problem in a workshop has become obsolete [BAK 74]: *set up times for operations are independent of the sequence and are included in processing times*.

The scheduling problem in a workshop considering tooling constraints is described in more detail below. The main elements are a series of machines and a collection of types of products to be produced on these machines. A process plan is associated with each product type and consists of a sequence of operations; each operation is characterized by a type of machine on which it must be performed, its processing time and the tools needed for each machine. The main hypotheses are as follows [BAK 74, WID 91a]:

– $m$ different machines are continuously available (no breakdown and no maintenance operation are considered);

– a set of $n$ type products is available for processing at time zero (each product type requires $m$ operations and each operation is processed on a different machine);

– each machine has a tool magazine with a limited capacity;

– a machine can process only one product at a time;

– a product can only be produced by one machine at a time;

– individual operations are non-preemptive;

– each individual operation requires a number of tools, which never exceeds the capacity of the tool magazine;

– process plans are known in advance.

The goal is to find a sequence of operations for each machine to minimize the makespan which is defined as the maximum of the completion times of all operations.

This problem, *if tooling constraints are not taken into consideration*, is the well known job shop scheduling problem. It is NP-complete, as demonstrated by Lawler, *et al.* [LAW 89]. A reader interested in a good summary of methods implemented to solve the job shop scheduling problem without tooling constraints should refer to work from Blazewicz, Domschke and Pesch [BLA 96].

Returning now to our problem with tooling constraints, because of the limited capacity of the tool magazines on the machines, tool switches are unavoidable and further complicate the problem of finding an optimal job shop scheduling [BLA 94]. When a *tool switch* happens, the machine's tool magazine is emptied of its tools, either partially or completely, and replenished with tools necessary for the production of the next products in the planned sequence. The time required for this change is usually significant. To illustrate this, we will use the following example: we assume that products A, B and C must be manufactured (in that order) on a machine with a tool magazine having a capacity of 10. These products require respectively 4, 5 and 3 tools different from one another. A tool switch must therefore happen after the manufacturing of products A and B.

In certain cases, different types of products may require common tools. These tools should not be duplicated in the tool store, thus free space may be preserved. In our example, if A and B use 2 of common tools, it is possible to include tools required for the production of the three products in the store without exceeding its capacity ($4 + 5 – 2 + 3 = 10$). In these conditions, it is not necessary to execute a tool switch.

Before moving on to the resolution of the job shop scheduling problem with tooling constraints with a tabu search, we will discuss a traditional case model in order to draw some lessons from it.

## 3.5.1. *Traditional job shop modeling*

For clarity purposes, from now on, the traditional job shop scheduling problem will be noted as JP, while JPT will be the job shop scheduling problem with tooling constraints.

Let $\mathbf{O} = \{1, ..., n\}$ denote the total set of operations which are to be performed. For each operation $i \in \mathbf{O}$ we denote:

– $P_i$ the product to which $i$ belongs;

– $M_i$ the machine on which $i$ is processed;

– $p_i$ the processing time of $i$;

– PP($i$) the operation preceding $i$ in the process plan of $P_i$ ;

– FP($i$) the operation following $i$ in the process plan of $P_i$ .

A job shop scheduling problem can be represented as a graph theoretical problem [ROY 64]. For a given instance of JP, the following graph G = (X, U, D) can be associated with it:

– $X = \{0, ..., n+1\} = \mathbf{O} \cup \{0, n+1\}$, where 0 and n + 1 are special vertices identifying the start and the completion of the schedule;

– $U = \{[i, j] \mid 1 \leq i, j \leq n, M_i = M_j \text{ and } P_i \neq P_j\}$.

– $D = \{(i, j) \mid 1 \leq i, j \leq n \text{ and } j = FP(i)\}$
$\cup \{(0, i) \mid 1 \leq i \leq n \text{ and } i \text{ is the initial operation for product } P_i\}$
$\cup \{(i, n+1) \mid 1 \leq i \leq n \text{ and } i \text{ is the final operation for product } P_i\}$

To each arc $(i,j) \in D$ or edge $[i, j] \in U$ with $i \geq 1$, we associate a length representing the duration $p_i$ of operation $i$. All arcs from vertex 0 have a length equal to zero. In other words, all arcs in D represent the different process plans. An orientation of the edges in U is called *feasible* if it does not create a circuit in G. A feasible orientation of the edge set U corresponds to a *feasible ordering* of the operations on the machines. The job shop scheduling problem is to find a feasible orientation of the edge set U in such a way that the longest route from 0 to $n + 1$ is of minimal length. Two operations requiring the same machine are defined as *adjacent* if the ending time of the first product is equal to the starting time of the second product on this same machine. An operation is called *critical* if it belongs to the longest route from 0 to $n + 1$. A *block* is a maximal sequence of adjacent operations which must be executed on a single machine and belonging to a longest route from 0 to $n + 1$.

The first adaptation of tabu search to the JP was developed by Taillard [TAI 94]. The set X of solutions is defined as the set of feasible orientations. Given a solution $s$ in X, a neighbor solution $s' \in N(s)$ is obtained by permuting two consecutive critical operations using the same machine. When two consecutive critical operations $i$ and $j$ are permuted, operation $j$ is introduced in the tabu list T: it is forbidden to permute $j$ with the next operation on machine $M_j$ during $|T|$ iterations. The neighborhood structure $N(s)$ used by Taillard has the following properties, demonstrated by van Laarhoven *et al.* [VAN 92]:

– if $s$ is a feasible orientation, then all elements in $N(s)$ are also feasible orientations;

– let **G** be the state space graph induced by X and $N(s)$ (see section 3.4.2) : for each solution $s \in X$, there is at least one path in **G** linking $s$ to an optimal solution $s^*$.

A second tabu search adaptation to the JP was proposed by Dell'Amico and Trubian [DEL 93]. Their method seems to dominate Taillard's approach. The main difference is the definition of neighborhood $N(s)$: for each operation $i$ in a block, they consider as neighbors of $s$ those solutions that can be generated by moving $i$ to the first or last position of the block to which it belongs, if the corresponding ordering is feasible. If that is not the case, operation *i* is moved to the position inside the block closest to the first or last position so that feasibility is preserved. Dell'Amico and Trubian have proven that this type of neighborhood leads to a state space graph **G** in which it is possible to reach an optimal solution starting from any initial solution [DEL 93].

Nowicki and Smutnicki [NOW 96] have developed a third adaptation of tabu search to the JP which is based, once again, on a different definition of the neighborhood N(*s*). These three adaptations of tabu search to the JP unfortunately do not consider tooling constraints.

### 3.5.2. *Comparing both types of problems*

As mentioned in the introduction, the time required to switch tools on a machine is significant. The moment where the contents of a tool magazine is modified will be called a *switching instant*. It lasts $\alpha + \beta r$ time units, where $\alpha$ is a fixed time due to the removal of the tool magazine from the machine, $\beta$ is a fixed time for each tool replacement and $r$ is the total number of tools to be replaced by other tools. In addition, a complete set of tools is attached to each machine (in this way, two machines can use identical tools simultaneously).

With these basic hypotheses, a comparative analysis of JP and JPT problems was conducted to determine how ingredients used in solution methods for the JP can also be used for the JPT . The following observations were made [HER 95, HER 96]:

– an optimal ordering of the operations on the machines for the JP does not necessarily correspond to an optimal ordering for the JPT;

– given an ordering of the operations, minimizing the makespan is not necessarily achieved by minimizing the number of switching instants;

– given an ordering of the operations, a schedule of minimum time for the JP can easily be obtained by searching for a longest path in a graph; this problem can be solved in polynomial time (see for example the Bellman algorithm adaptation described in [TAI 94] or the O*(n)* algorithm developed by Adams *et al.* [ADA 88]). For the JPT problem, the complexity of this problem is still open. In other words, given an ordering of the operations on the machines, we do not know how to take into account tooling constraints for minimizing the makespan;

– assuming that the number of switching instants is given for each machine, it is not always optimal to plan the switching instants to the earliest or latest;

– if we know the sequence of operations on each machine as well as the contents of the tool magazine, then a minimum time scheduling can be found by solving a longest path problem: in this case, switching instants are considered as additional operations whose duration is known and the graph is constructed in the same way as for the JP.

As mentioned above, when solving the JP, neighborhood structures from Taillard and from Dell'Amico and Trubian generate a state space graph **G** in which it is possible to reach an optimal solution from any initial solution. This is unfortunately not the case with the JPT. In fact, by using these neighborhood structures, it is possible to "cycle" between a set of solutions while the optimal solution may be elsewhere (the state space graph **G** contains several connected components) [HER 96].

### 3.5.3. *Tool switching*

As previously mentioned, despite knowing operation sequences on machines, it is still difficult to determine the best way to take tooling constraints into consideration. A possibility is to use the heuristic method proposed in [HER 96]. It contains three phases which are summarized below.

In the first phase, all machines are treated independently. We want to evaluate the *number of switching instants*. We determine a set of operations which immediately precede a switching instant. The INSTANT algorithm considers the ordered set $\{o_1, ..., o_r\}$ of operations on a machine $k$ ($1 \leq k \leq m$) and sequentially schedules the switching instants only when forced to. Hence, if a switching instant precedes an operation $o_i$ ($1 < i \leq r$), this means that the set of tools required to complete $o_i$ on $k$ cannot be added to the tool magazine without exceeding its capacity. The number of switching instants determined by this algorithm is minimal. In the following, we will only consider schedules having that number of switching instants.

In the second heuristic phase, we want to determine the *contents of the tool magazines*. As in the first phase, all machines are treated independently. Since tool replacements can only occur during the switching instants, all operations between two consecutive switching instants can be considered as a unique operation requiring a known set of tools to be processed. Thus, given an ordered set of operations which must be processed on a machine $k$ ($1 \leq k \leq m$), we must minimize the total number of tool switches. It was proven by Tang and Denardo [TAN 88] that given an ordered set of operations which must be performed on a unique machine, minimizing the total number of tool switches is a problem that can be solved in an exact way and in a reasonable amount of time. The REPLACEMENT algorithm uses the KTNS (keep tool needed soonest) policy proposed by Tang and Denardo [TAN 88]. This replacement policy has the following properties:

– at any given instant, no tool is introduced unless it is required for the next operation;

– if a tool must be inserted, the tools remaining (not taken out) are those needed the soonest.

Once the tool switching problem is solved on each machine, a solution to the JPT can be obtained by solving a longest path problem. In fact, all switching instants can be considered as additional operations with known duration, and a graph can then be constructed in the same way as for the JP. The two first heuristic phases consider the machines independently. In the third phase, we treat all the machines simultaneously. We try to *improve a solution* by scheduling a switching moment on a machine when no product is ready to be processed on it. An algorithm, which we call IMPROVE, is proposed in [HER 96] to improve the schedule of the switching instants.

### 3.5.4. *TOMATO algorithm*

We can now describe a tabu search algorithm for the job shop scheduling problem with tooling constraints.

Let *s'* be a neighbor solution of *s* obtained by moving an operation on a machine *k*. To evaluate *s'*, we first determine the operations on *k* which immediately precede a switching instant with the help of the INSTANT algorithm. We then apply the REPLACEMENT algorithm on *k* and compute the makespan of *s'* by solving a longest path problem (see section 2.4.2). Once the decision is made to move from a current solution *s* to a neighbor solution $s' \in N(s)$, we try to improve the schedule of switching instants in *s'* by using the IMPROVE algorithm.

The tabu T list is a set of operations not authorized to be moved during a certain number of iterations. When an operation *i* is moved before or after an operation *j* on a machine *k*, operation *i* is introduced in the tabu list T. In addition, if *j* is an immediate predecessor or successor to *i*, the operation *j* is also introduced in the tabu T: indeed, in that case, *s* could be obtained from *s'* by moving *j* before of after *i*. The length of the tabu list is randomly chosen at each iteration in the interval $[n, \lfloor 3n/2 \rfloor]$, following a uniform distribution.

The objective function *f* is the makespan (the time where the last job is completed). Finally, the iterative process ends when *nbmax* iterations are completed without improvement of $f(s^*)$. An algorithmic form of this tabu search adaptation to the job shop scheduling problem with tooling constraints is presented in Algorithm 3.8. The method was named TOMATO (for TOol MAnagement with Tabu Optimization) in [HER 96].

```
Initialization
    find an initial solution for the JP;
    let s be the schedule obtained by using the INSTANT and REPLACEMENT heuristics
    on each machine, and by solving a longest path problem;
    improve s by using the IMPROVE procedure;
    set nbiter $\leftarrow 0$, $T \leftarrow \varnothing$, $s^* \leftarrow s$ and best_iter $\leftarrow 0$;
Iterative process
while (nbiter-best_iter<nbmax) do
    set nbiter $\leftarrow$ nbiter+1;
    evaluate each solution s' of N(s) by using the INSTANT and REPLACEMENT
    procedures for the machine on which an operation is moved, and by solving a longest
    path problem;
    choose the best solution s' of N(s) so that $f(s') < f(s^*)$ or s' is not tabu;
    improve s' by using the IMPROVE procedure;
    update the list T of tabu movements;
    set $s \leftarrow s'$;
    if $f(s) < f(s^*)$ then set $s^* \leftarrow s$ and best_iter $\leftarrow$ nbiter
```

**Algorithm 3.8.** *The TOMATO algorithm*

TOMATO should only be applied to job shop scheduling problems in which the tool magazines are of relatively relatively limited capacity, which is the case with small to medium sized companies. When the capacity of the tool magazines increases and becomes far greater than the average number of tools necessary for the execution of an operation, the problem generally becomes similar to the job shop scheduling problem without tooling constraints. In this case, a solution can be obtained with the method proposed by Dell'Amico and Trubian, and (the few) switching instants can then be planned by using the INSTANT, REPLACEMENT and IMPROVE algorithms. TOMATO is a flexible method that can be used by the production manager who wants either a quick but not precise estimation of the makespan or an accurate solution. The desired trade-off between computational time and solution quality can be established by adjusting the parameter *nbmax*. TOMATO can be used as a descent method by setting *nbmax* = 0. It was observed in [HER 96] that TOMATO provides in less than 1 second solutions whose value are approximately only 5% above the best known solution values.

## 3.6. Conclusion

Upon reaching the end of this chapter, a reader inevitably asks the following question: which is the best metaheuristic?

Unfortunately, a direct and specific answer is impossible: even though some similarities are obvious between these different approaches (e.g., the use of neighborhoods), they are different in sensitive areas (simulated annealing uses an energy function, tabu search handles a list of prohibited movements, genetic algorithms have crossover operators, etc.). Comparisons are difficult to make for two reasons: on the one hand, such a comparison requires a fine tuning of all parameters of all methods, and on the other hand, the quality of the solutions produced by a metaheuristic depends on the available computing time. However, regardless of the metaheuristic used, it is now a fact that all these algorithms are the only effective solution methods for many large size combinatorial optimization problems. The promising results of hybrid approaches bode well for vital progress in this field.

In conclusion, we mention that the reader wanting to have a deeper understanding of metaheuristics for general combinatorial optimization problems can refer to [REE 95, DRE 03]. Also, successful adaptations of metaheuristics for the solution of real problems are described in [IBA 05].

## 3.7. Bibliography

[AAR 89] AARTS E.H.L. and KORST J., *Simulated annealing and Boltzmann machines*, John Wiley, New York, 1989.

[ADA 88] ADAMS J., BALAS E. and ZAWACK D., "The shifting bottleneck procedure for job shop scheduling", *Management Science*, vol. 34, p. 391-401, 1988.

[BAK 74] BAKER K.R., *Introduction to sequencing and scheduling*, John Wiley, New York, 1974.

[BLA 94] BLAZEWICZ J. and FINKE G., "Scheduling with resource management in manufacturing systems", *European Journal of Operational Research,* vol. 76, p. 1-14, 1994.

[BLA 96] BLAZEWICZ J., DOMSCHKE W. and PESCH E., "The job shop scheduling problem: conventional and new solution techniques", *European Journal of Operational Research,* vol. 93, p. 1-33, 1996.

[CAL 99] CALEGARI P., CORAY G., HERTZ A., KOBLER D. and KUONEN P., "A taxonomy of evolutionary algorithms in combinatorial optimization", *Journal of Heuristics*, vol. 5, p. 145-158, 1999.

[CAR 88] CARLIER J. and CHRÉTIENNE P., *Problèmes d'ordonnancement*, Masson, 1988.

[CER 85] CERNY V., “Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm”, *Journal of Optimization Theory and Applications*, vol. 45, p. 41-51, 1985.

[COL 88] COLLINS N.E. “Simulated annealing: an annotated bibliography”, *American Journal of Mathematical and Management Sciences*, vol. 8, p. 209-307, 1988.

[COL 91] COLORNI A., DORIGO M. and MANIEZZO V., “Distributed optimization by ant colonies”, *First European Conference on Artificial Life*, Paris, p 134-142, 1991.

[COL 92] COLORNI A., DORIGO M. and MANIEZZO V., “An investigation of some properties of an ant algorithm”, *Second Conference on Parallel Problem Solving from Nature*, Brussels, p. 509-520, 1992.

[COL 94] COLORNI A., DORIGO M., MANIEZZO V. and TRUBIAN M., “Ant system for job shop scheduling”, *Belgian Journal of Operations Research, Statistics and Computer Science*, vol. 34, p. 39-53, 1994.

[CON 88] Committee on the Next Decade of Operations Research (CONDOR), “Operations research: the next decade”, *Operations Research* , vol. 36, p. 619-637, 1988.

[COS 95a] COSTA D., “An evolutionary tabu search algorithm and the NHL scheduling problem”, *INFOR*, vol. 33, p. 161-178, 1995.

[COS 95b] COSTA D., HERTZ A. and DUBUIS O., “Embedding of sequential procedures within an evolutionary algorithm for coloring problems in graphs”, *Journal of Heuristics*, vol. 1, p. 105-128, 1995.

[COS 95c] COSTA D., Méthodes de résolution constructives, séquentielles et évolutives pour des problèmes d’affectation sous contraintes, Thesis no. 1411, Department of Mathematics, Ecole Polytechnique Fédérale de Lausanne, 1995.

[COS 97] COSTA D. and HERTZ A., “Ants can colour graphs”, *Journal of the Operational Research Society*, vol. 48, p. 295-305, 1997.

[CUN 97] CUNG V.-D., MAUTOR T., MICHELON P. and TAVARES A., “A Scatter search based approach for the quadratic assignment problem”, *IEEE International Conference on Evolutionary Computation*, p. 165-170, 1997.

[DEL 93] DELL’AMICO M. and TRUBIAN M., “Applying tabu search to the job-shop scheduling problem”, *Annals of Operations Research*, vol. 41, p. 231-252, 1993.

[DOR 96] DORIGO M., MANIEZZO V. and COLORNI A., “The ant system: optimization by a colony of cooperating agents”, *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 26, p. 29-41, 1996.

[DOR 04] DORIGO M. and STÜTZLE T., *Ant Colony Optimization*, MIT Press, Cambridge, 2004.

[DRE 03] DRÉO J., PÉTROWSKI A., SIARRY P. and TAILLARD É., *Métaheuristiques pour l’optimisation difficile*, Éditions Eyrolles, Paris, 2003.

[DUE 90] DUECK G. and SCHEUER T., “Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing”, *Journal of Computational Physics*, vol. 90, p. 161-175, 1990.

[DUE 93] DUECK G., “New optimization heuristics: the great deluge algorithm and the record-to-record travel”, *Journal of Computational Physics*, vol. 104, p. 86-92, 1993.

[EGL 90] EGLESE R.W., “Simulated annealing: a tool for operational research”, *European Journal of Operational Research*, vol. 46, p. 271-281, 1990.

[FAI 92] FAIGLE U. and KERN W., “Some convergence results for probabilistic tabu search”, *ORSA Journal on Computing*, vol. 4, p. 32-37, 1992.

[FAL 96] FALKENAUER E., “A hybrid grouping genetic algorithm for bin packing”, *Journal of Heuristics*, vol. 2, p. 5-30, 1996.

[FER 96] FERLAND J.A., HERTZ A. and LAVOIE A., “An object oriented methodology for solving assignment type problems with neighborhood search techniques”, *Operations Research*, vol. 44, p. 347-359, 1996.

[FLE 94] FLEURENT C., Algorithmes génétiques hybrides pour l’optimisation combinatoire, Thesis, Department of Information Technology and Operations Research, University of Montreal, 1994.

[FLE 96a] FLEURENT C., GLOVER F., MICHELON P. and VALLI Z., “A scatter search approach for unconstrained continuous optimization”, *IEEE International Conference on Evolutionary Computation*, p. 643-648, 1996.

[FLE 96b] FLEURENT C. and FERLAND J.A., “Genetic and hybrid algorithms for graph coloring”, *Annals of Operations Research*, vol. 63, p. 437-461, 1996.

[GAL 06] GALINIER P. and HERTZ A., “A Survey of Local Search Methods for Graph Coloring”, *Computers and Operations Research*, vol. 33, p. 2547-2562, 2006.

[GAR 79] GAREY M.R. and JOHNSON D.S., *Computers and intractability: a guide to the theory of NP-completeness*, Freeman, 1979.

[GEM 84] GEMAN S. and GEMAN D., “Stochastic relaxation, Gibbs distributions, and bayesian restoration of images”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, p. 721-741, 1984.

[GLA 96] GLASS C.A. and POTTS C.N., “A comparison of local search methods for flow shop scheduling”, *Annals of Operations Research*, vol. 63, p. 489-509, 1996.

[GLO 77] GLOVER F., “Heuristics for integer programming using surrogate constraints”, *Decision Sciences*, vol. 8, p. 156-166, 1977.

[GLO 86] GLOVER F., “Future paths for integer programming and links to artificial intelligence”, *Computers and Operations Research*, vol. 13, p. 533-549, 1986.

[GLO 89] GLOVER F., “Tabu search: Part I”, *ORSA Journal on Computing*, vol. 1, p. 190-206, 1989.

[GLO 90] GLOVER F., “Tabu search: Part II”, *ORSA Journal on Computing*, vol. 2, p. 4-32, 1990.

[GLO 93a] GLOVER F., TAILLARD E., LAGUNA M. and DE WERRA D. (Eds.), “Tabu search”, *Annals of Operations Research*, vol. 41, 1993.

[GLO 93b] GLOVER F., TAILLARD E. and DE WERRA D., "A user's guide to tabu search", *Annals of Operations Research,* vol. 41, p. 3-28, 1993.

[GLO 94] GLOVER F., "Genetic algorithms and scatter search: unsuspected potentials", *Statistics and Computing*, vol. 4, p. 131-140, 1994.

[GLO 95] GLOVER F., "Scatter search and star-paths: beyond the genetic metaphor", *OR Spektrum*, vol. 17, p. 125-138, 1995.

[GLO 97] GLOVER F. and LAGUNA M., *Tabu search*, Kluwer Academic Publishers, Norwell, 1997.

[GOL 87] GOLDBERG D.E., *Genetic algorithms in search, optimization, and machine learning*, Addison Wesley, 1987.

[GON 85] GONDRAN M. and MINOUX M., *Graphes et algorithmes*, Eyrolles Publishers, Paris, 1985.

[GRE 87] GREFENSTETTE J.J., "Incorporating problem specific knowledge into genetic algorithms", in *Genetic Algorithms and Simulated Annealing*, Davis. L. (ed.), Morgan Kaufmann Publishers, p. 42-60, 1987.

[HAJ 88] HAJEK B., "Cooling schedules for optimal annealing", *Mathematics of Operations Research*, vol. 13, p. 311-329, 1988.

[HAN 86] HANSEN P., "The steepest ascent mildest descent heuristic for combinatorial programming", *Congress on Numerical Methods in Combinatorial Optimization*, Capri, Italy, 1986.

[HER 95] HERTZ A. and WIDMER M., "La méthode TABOU appliquée aux problèmes d'ordonnancement", *RAIRO/Automatique, Productique, Informatique Industrielle*, vol. 29, p. 353-378, 1995.

[HER 96] HERTZ A. and WIDMER M., "An improved tabu search approach for solving the job shop scheduling problem with tooling constraints", *Discrete Applied Mathematics*, vol. 65, p. 319-346, 1996.

[HER 97] HERTZ A., TAILLARD E. and DE WERRA D., "Tabu search", in *Local search in combinatorial optimization,* E. Aarts and J.K. Lenstra (eds.), John Wiley, 1997.

[HER 03] HERTZ A. and WIDMER M., "Guidelines for the use of meta-heuristics in combinatorial optimization", *European Journal of Operational Research*, vol. 151, p. 247-252, 2003.

[HOL 75] HOLLAND J.H., *Adaptation in natural and artificial systems*, University of Michigan Press, Ann Arbor, 1975.

[IBA 05] IBARAKI T., NONOBE K. and YAGIURA M., *Métaheuristics : progress as real problem solvers*, Springer, 2005.

[KEL 96] KELLY J., RANGASWAMY B. and XU J., "A scatter search-based learning algorithm for neural network training", *Journal of Heuristics*, vol. 2, p. 129-146, 1996.

[KIR 83] KIRKPATRICK S., GELATT C.D. Jr. and VECCHI M.P., "Optimization by simulated annealing", *Science*, vol. 220, p. 671-680, 1983.

[LAW 89] LAWLER E.L., LENSTRA J.K., RINNOOY KAN A.H.G. and SHMOYS D.B., "Sequencing and scheduling: algorithms and complexity", Chapter 9 in *Logistics of production and inventory*, Graves S.C., Rinnooy Kan A.H.G. and Zipkin P.H. (Eds.), Elsevier Science Publishers, p. 445-522, 1993.

[MAN 99] MANIEZZO V. and COLORNI A., "The ant system applied to the quadratic assignment problem", *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, p. 769-778, 1999.

[MET 53] METROPOLIS N., ROSENBLUTH A., ROSENBLUTH M., TELLER A. and TELLER E., "Equation of state calculations by fast computing machines", *Journal of Chemical Physics*, vol. 21, p. 1087-1092, 1953.

[MOS 93] MOSCATO P., "An introduction to population approaches for optimization and hierarchical objective functions: a discussion on the role of tabu search", *Annals of Operations Research*, vol. 41, p. 85-121, 1993.

[MÜH 88] MÜHLENBEIN H., GORGES-SCHLEUTER M. and KRÄMER O., "Evolution algorithms in combinatorial optimization", *Parallel Computing*, vol. 7, 65-85, 1988.

[MÜL 81] MÜLLER-MERBACH H., "Heuristics and their design: a survey", *European Journal of Operational Research*, vol. 8, p. 1-23, 1981.

[NAW 83] NAWAZ M., ENSCORE JR. E.E. and HAM I., "A heuristic algorithm for the m-machine, n-job flow shop sequencing problem", *Omega*, vol. 11, p. 91-95, 1983.

[NIC 71] NICHOLSON T.A.J., *Optimization in industry*, *vol. 1 : Optimization Techniques*, Longmann Press, London, 1971.

[NOW 96] NOWICKI E. and SMUTNICKI C., "A fast tabu search algorithm for the job-shop problem", *Management Science*, vol. 42, p. 797-813, 1996.

[OSM 94] OSMAN I.H. and CHRISTOFIDES N., "Capacitated clustering problems by hybrid simulated annealing and tabu search", *International Transactions in Operational Research*, vol. 1, p. 317-336, 1994.

[RAD 94] RADCLIFFE N.J. and SURRY P.D., "Formal memetic algorithms", *Lecture Notes in Computer Science*, vol. 865, p. 1-16, 1994

[REE 93] REEVES C.R., *Modern heuristic techniques for combinatorial optimization*, Blackwell Scientific Publications, Oxford, 1993.

[REE 95] REEVES C.R., *Modern heuristic techniques for combinatorial problems*, McGraw-Hill, 1995.

[REE 96] REEVES C., "Hybrid genetic algorithms for bin-packing and related problems", *Annals of Operations Research*, vol. 63, p. 371-396, 1996.

[ROY 64] ROY B. and SUSSMANN B., Les problèmes d'ordonnancement avec contraintes disjonctives, Technical report, SEMA, Montrouge, 1964.

[RUT 89] RUTENBAR R. A., "Simulated annealing algorithms: an overview", *IEEE Circuits and Devices Magazine*, vol. 5, p. 19-26, 1989.

[SIL 80] SILVER E.A., VIDAL R.V. and DE WERRA D., "A tutorial on heuristic methods", *European Journal of Operational Research*, vol. 5, p. 153-162, 1980.

[SIN 93] SINCLAIR M., "Comparison of the performance of modern heuristics for combinatorial optimization on real data", *Computers and Operations Research*, vol. 20, p. 687-695, 1993.

[SOR 94] SORIANO P., Applications de la méthode de recherche avec tabous à divers problèmes d'optimisation combinatoire, Thesis, Department of Information Technology and Operations Research, University of Montreal, 1994.

[TAI 93] TAILLARD E., Recherches itératives dirigées parallèles, Thesis no. 1153, Department of Mathematics, Ecole Polytechnique Fédérale de Lausanne, 1993.

[TAI 94] TAILLARD E., "Parallel taboo search technique for the job shop scheduling problem", *ORSA Journal on Computing*, vol. 6, p. 108-117, 1994.

[TAI 95] TAILLARD E., "Comparison of iterative searches for the quadratic assignment problem", *Location Science*, vol. 3, p. 87-105, 1995.

[TAN 88] TANG C.S. and DENARDO V., "Models arising from a flexible manufacturing machine, Part I: Minimization of the number of tool switches, Part II: Minimization of the number of switching moments", *Operations Research*, vol. 36, p. 778-784, 1988.

[THE 94] THERAULAZ G., BONABEAU E., GOSS S. and DENEUBOURG J.L., "L'intelligence collective", *Pour la Science*, no. 128, 1994.

[VAN 87] VAN LAARHOVEN P.J.M. and AARTS E.H.L., *Simulated annealing: theory and applications*, D. Reidel Publishing Company, Dordrecht, 1987.

[VAN 92] VAN LAARHOVEN P.J.M., AARTS E.H.L. and LENSTRA J.K., "Job shop scheduling by simulated annealing", *Operations Research*, vol. 40, p. 113-125, 1992.

[VID 93] VIDAL R.V. (ed.), *Applied simulated annealing*, Lecture Notes in Economics and Mathematical Systems 396, Springer-Verlag, Berlin, 1993.

[WER 89] DE WERRA D. and HERTZ A., "Tabu search techniques: a tutorial and an application to neural networks", *OR Spektrum*, vol. 11, p. 131-141, 1989.

[WID 91a] WIDMER M., "Job shop scheduling with tooling constraints: a tabu search approach", *Journal of the Operational Research Society*, vol. 42, p. 75-82, 1991.

[WID 91b] WIDMER M., *Modèles mathématiques pour une gestion efficace des ateliers flexibles*, Presses Polytechniques and Universitaires Romandes, 1991.

[WID 98] WIDMER M., Organisation industrielle: le réel apport des mathématiques, Habilitation thesis, Fribourg University, 1998.

[ZAN 89] ZANAKIS H.S., EVANS J.R. and VAZACOPOULOS A.A., "Heuristic methods and applications: a categorized survey", *European Journal of Operational Research*, vol. 43, p. 88-110, 1989.