- **Accepting nonimproving neighbors:** These approaches enable moves that degrade the current solution. It becomes possible to move out the basin of attraction of a given local optimum. Simulated annealing and tabu search are popular representative of this class of algorithms. Simulated annealing was the first algorithm addressing explicitly the question "why should we consider only downhill moves?"
- **Changing the neighborhood:** This class of approaches consists in changing the neighborhood structure during the search. For instance, this approach is used in variable neighborhood search strategies.
- **Changing the objective function or the input data of the problem:** In this class, the problem is transformed by perturbing the input data of the problem, the objective function or the constraints, in the hope to solve more efficiently the original problem. This approach has been implemented in the guided local search, the smoothing strategies, and the noising methods. The two last approaches may be viewed as approaches changing the landscape of the problem to solve.

2.4 SIMULATED ANNEALING

Simulated annealing applied to optimization problems emerges from the work of S. Kirkpatrick et al. [464] and V. Cerny [114]. In these pioneering works, SA has been applied to graph partitioning and VLSI design. In the 1980s, SA had a major impact on the field of heuristic search for its simplicity and efficiency in solving combinatorial optimization problems. Then, it has been extended to deal with continuous optimization problems [204,512,596].

SA is based on the principles of statistical mechanics whereby the annealing process requires heating and then slowly cooling a substance to obtain a strong crystalline structure. The strength of the structure depends on the rate of cooling metals. If the initial temperature is not sufficiently high or a fast cooling is applied, imperfections (metastable states) are obtained. In this case, the cooling solid will not attain thermal equilibrium at each temperature. Strong crystals are grown from careful and slow cooling. The SA algorithm simulates the energy changes in a system subjected to a cooling process until it converges to an equilibrium state (steady frozen state). This scheme was developed in 1953 by Metropolis [543].

Table 2.4 illustrates the analogy between the physical system and the optimization problem. The objective function of the problem is analogous to the energy state of the system. A solution of the optimization problem corresponds to a system state. The decision variables associated with a solution of the problem are analogous to the molecular positions. The global optimum corresponds to the ground state of the system. Finding a local minimum implies that a metastable state has been reached.

SA is a stochastic algorithm that enables under some conditions the degradation of a solution. The objective is to escape from local optima and so to delay the convergence. SA is a memoryless algorithm in the sense that the algorithm does not

TABLE 2.4 Analogy Between the Physical System and the Optimization Problem

Physical System	Optimization Problem
System state	Solution
Molecular positions	Decision variables
Energy	Objective function
Ground state	Global optimal solution
Metastable state	Local optimum
Rapid quenching	Local search
Temperature	Control parameter T
Careful annealing	Simulated annealing

use any information gathered during the search. From an initial solution, SA proceeds in several iterations. At each iteration, a random neighbor is generated. Moves that improve the cost function are always accepted. Otherwise, the neighbor is selected with a given probability that depends on the current temperature and the amount of degradation ΔE of the objective function. ΔE represents the difference in the objective value (energy) between the current solution and the generated neighboring solution. As the algorithm progresses, the probability that such moves are accepted decreases (Fig. 2.25). This probability follows, in general, the Boltzmann distribution:

$$P(\Delta E, T) = \mathrm{e}^{-\frac{f(s')-f(s)}{T}}$$

It uses a control parameter, called temperature, to determine the probability of accepting nonimproving solutions. At a particular level of temperature, many trials are explored. Once an equilibrium state is reached, the temperature is gradually decreased

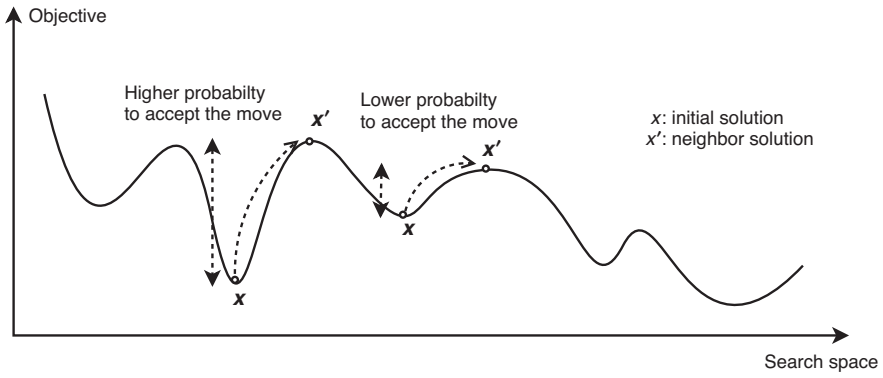

FIGURE 2.25 Simulated annealing escaping from local optima. The higher the temperature, the more significant the probability of accepting a worst move. At a given temperature, the lower the increase of the objective function, the more significant the probability of accepting the move. A better move is always accepted.

according to a cooling schedule such that few nonimproving solutions are accepted at the end of the search. Algorithm 2.3 describes the template of the SA algorithm.

Algorithm 2.3 Template of simulated annealing algorithm.

Input: Cooling schedule.
$s = s_0$; /* Generation of the initial solution */
$T = T_{max}$; /* Starting temperature */
Repeat
  Repeat /* At a fixed temperature */
    Generate a random neighbor s' ;
    $\Delta E = f(s') - f(s)$;
    If $\Delta E \leq 0$ **Then** $s = s'$ /* Accept the neighbor solution */
    Else Accept s' with a probability $e^{\frac{-\Delta E}{T}}$;
  Until Equilibrium condition
  /* e.g. a given number of iterations executed at each temperature T */
  $T = g(T)$; /* Temperature update */
Until Stopping criteria satisfied /* e.g. $T < T_{min}$ */
Output: Best solution found.

Example 2.23 Illustration of the SA algorithm. Let us maximize the continuous function $f(x) = x^3 - 60x^2 + 900x + 100$. A solution x is represented as a string of 5 bits. The neighborhood consists in flipping randomly a bit. The global maximum of this function is 01010 ($x = 10$, $f(x) = 4100$). The first scenario starts from the solution 10011 ($x = 19$, $f(x) = 2399$) with an initial temperature T_0 equal to 500 (Table 2.5). The second scenario starts from the same solution 10011 with an initial temperature T_0 equal to 100 (Table 2.6). The initial temperature is not high enough and the algorithm gets stuck by local optima.

In addition to the current solution, the best solution found since the beginning of the search is stored. Few parameters control the progress of the search, which are the temperature and the number of iterations performed at each temperature.

Theoretical analysis of the asymptotic convergence of SA is well developed [478]. The search may be modeled by a Markov chain, where the next state depends only

TABLE 2.5 First Scenario $T = 500$ and Initial Solution (10011)

T	Move	Solution	f	Δf	Move?	New Neighbor Solution
500	1	00011	2287	112	Yes	00011
450	3	00111	3803	<0	Yes	00111
405	5	00110	3556	247	Yes	00110
364.5	2	01110	3684	<0	Yes	01110
328	4	01100	3998	<0	Yes	01100
295.2	3	01000	3972	16	Yes	01000
265.7	4	01010	**4100**	<0	Yes	01010
239.1	5	01011	4071	29	Yes	01011
215.2	1	11011	343	3728	No	01011

TABLE 2.6 Second Scenario: T = 100 and Initial Solution (10011). When Temperature is not High Enough, Algorithm Gets Stuck

T	Move	Solution	f	Δf	Move?	New Neighbor Solution
100	1	00011	2287	112	No	10011
90	3	10111	1227	1172	No	10011
81	5	10010	2692	<0	Yes	10010
72.9	2	11010	516	2176	No	10010
65.6	4	10000	**3236**	<0	Yes	10000
59	3	10100	2100	1136	Yes	10000

on the current state. There is a guarantee of convergence toward the optimal solution:

$$\Pr(s_M \in R) \to 1 \quad \text{as} \quad M \to \infty$$

where R represents the set of global optimal solutions and s_M the solution at iteration M under the following slow cooling schedule:

$$T_k = \frac{\Gamma}{\log k}$$

where Γ is a constant. In practice, such a cooling schedule is useless because it is an asymptotic convergence; that is, the convergence is obtained after an infinite number of iterations. However, much more work is needed in the analysis of finite time performance [267].

In addition to the common design issues for S-metaheuristics such as the definition of the neighborhood and the generation of the initial solution, the main design issues specific to SA are

- **The acceptance probability function:** It is the main element of SA that enables nonimproving neighbors to be selected.
- **The cooling schedule:** The cooling schedule defines the temperature at each step of the algorithm. It has an essential role in the efficiency and the effectiveness of the algorithm.

The following sections present a practical guideline in the definition of the acceptance probability function and the cooling schedule in SA.

2.4.1 Move Acceptance

The system can escape from local optima due to the probabilistic acceptance of a nonimproving neighbor. The probability of accepting a nonimproving neighbor is proportional to the temperature T and inversely proportional to the change of the objective function ΔE.

The law of thermodynamics states that at temperature T, the probability of an increase in energy of magnitude, ΔE, is given by $P(\Delta E, T) = \exp(-\Delta E/kt)$ where k is a constant known as Boltzmann's constant. So, the acceptance probability of a nonimproving move is

$$P(\Delta E, T) = \exp\left(\frac{-\delta E}{kt}\right) > R$$

where ΔE is the change in the evaluation function, T is the current temperature, and R is a uniform random number between 0 and 1.

At high temperatures, the probability of accepting worse moves is high. If $T = \infty$, all moves are accepted, which corresponds to a random local walk in the landscape. At low temperatures, the probability of accepting worse moves decreases. If $T = 0$, no worse moves are accepted and the search is equivalent to local search (i.e., hill climbing). Moreover, the probability of accepting a large deterioration in solution quality decreases exponentially toward 0 according to the Boltzmann distribution.

2.4.2 Cooling Schedule

The cooling schedule defines for each step of the algorithm i the temperature T_i. It has a great impact on the success of the SA optimization algorithm. Indeed, the performance of SA is very sensitive to the choice of the cooling schedule.

The parameters to consider in defining a cooling schedule are the starting temperature, the equilibrium state, a cooling function, and the final temperature that defines the stopping criteria. A guideline dealing with the initialization of each parameter is given next.

2.4.2.1 Initial Temperature If the starting temperature is very high, the search will be more or less a random local search. Otherwise, if the initial temperature is very low, the search will be more or less a first improving local search algorithm. Hence, we have to balance between these two extreme procedures. The starting temperature must not be too high to conduct a random search for a period of time but high enough to allow moves to almost neighborhood state.

There are three main strategies that can be used to deal with this parameter:

- **Accept all:** The starting temperature is set high enough to accept all neighbors during the initial phase of the algorithm [464]. The main drawback of this strategy is its high computational cost.
- **Acceptance deviation:** The starting temperature is computed by $k\sigma$ using preliminary experimentations, where σ represents the standard deviation of difference between values of objective functions and $k = -3/\ln(p)$ with the acceptance probability of p, which is greater than 3σ [392].

- **Acceptance ratio:** The starting temperature is defined so as to make the acceptance ratio of solutions greater than a predetermined value a_0

$$T_0 = \frac{\Delta^+}{\ln(m_1(a_0 - 1)/m_2 + a_0)}$$

where m_1 and m_2 are the numbers of solutions to be decreased and increased in preliminary experiments, respectively, and Δ^+ is the average of objective function values increased [2]. For instance, the initial temperature should be initialized in such a way that the acceptance rate is in the interval [40%, 50%].

2.4.2.2 Equilibrium State To reach an equilibrium state at each temperature, a number of sufficient transitions (moves) must be applied. Theory suggests that the number of iterations at each temperature might be exponential to the problem size, which is a difficult strategy to apply in practice. The number of iterations must be set according to the size of the problem instance and particularly proportional to the neighborhood size $|N(s)|$. The number of transitions visited may be as follows:

- **Static:** In a static strategy, the number of transitions is determined before the search starts. For instance, a given proportion y of the neighborhood $N(s)$ is explored. Hence, the number of generated neighbors from a solution s is $y \cdot |N(s)|$. The more significant the ratio y, the higher the computational cost and the better the results.
- **Adaptive:** The number of generated neighbors will depend on the characteristics of the search. For instance, it is not necessary to reach the equilibrium state at each temperature. Nonequilibrium simulated annealing algorithms may be used: the cooling schedule may be enforced as soon as an improving neighbor solution is generated. This feature may result in the reduction of the computational time without compromising the quality of the obtained solutions [107].

  Another adaptive approach using both the worst and the best solutions found in the inner loop of the algorithm may be used. Let f_l (resp. f_h) denote the smallest (resp. largest) objective function value in the current inner loop. The next number of transitions L is defined as follows:

$$L = L_B + \lfloor L_B \cdot F_- \rfloor$$

where $\lfloor x \rfloor$ is the largest integer smaller than x, $F_- = 1 - \exp(-(f_h - f_l)/f_h)$, and L_B is the initial value of the number of transitions [25].

2.4.2.3 Cooling In the SA algorithm, the temperature is decreased gradually such that

$$T_i > 0, \forall i$$

and

$$\lim_{i \longrightarrow \infty} T_i = 0$$

There is always a compromise between the quality of the obtained solutions and the speed of the cooling schedule. If the temperature is decreased slowly, better solutions are obtained but with a more significant computation time. The temperature T can be updated in different ways:

- **Linear:** In the trivial linear schedule, the temperature T is updated as follows: $T = T - \beta$, where β is a specified constant value. Hence, we have

$$T_i = T_0 - i \times \beta$$

  where T_i represents the temperature at iteration i.
- **Geometric:** In the geometric schedule, the temperature is updated using the formula

$$T = \alpha T$$

  where $\alpha \in]0, 1[$. It is the most popular cooling function. Experience has shown that α should be between 0.5 and 0.99.
- **Logarithmic:** The following formula is used:

$$T_i = \frac{T_0}{\log(i)}$$

  This schedule is too slow to be applied in practice but has the property of the convergence proof to a global optimum [303].
- **Very slow decrease:** The main trade-off in a cooling schedule is the use of a large number of iterations at a few temperatures or a small number of iterations at many temperatures. A very slow cooling schedule such as

$$T_{i+1} = \frac{T_i}{1 + \beta T_i}$$

  may be used [521], where $\beta = T_0 - T_F/(L - 1)T_0T_F$ and T_F is the final temperature. Only one iteration is allowed at each temperature in this very slow decreasing function.
- **Nonmonotonic:** Typical cooling schedules use monotone temperatures. Some nonmonotone scheduling schemes where the temperature is increased again may be suggested [390]. This will encourage the diversification in the search space. For some types of search landscapes, the optimal schedule is nonmonotone [352].

- **Adaptive:** Most of the cooling schedules are static in the sense that the cooling schedule is defined completely *a priori*. In this case, the cooling schedule is "blind" to the characteristics of the search landscape. In an adaptive cooling schedule, the decreasing rate is dynamic and depends on some information obtained during the search [402]. A dynamic cooling schedule may be used where a small number of iterations are carried out at high temperatures and a large number of iterations at low temperatures.

2.4.2.4 Stopping Condition Concerning the stopping condition, theory suggests a final temperature equal to 0. In practice, one can stop the search when the probability of accepting a move is negligible. The following stopping criteria may be used:

- Reaching a final temperature T_F is the most popular stopping criteria. This temperature must be low (e.g., $T_{min} = 0.01$).
- Achieving a predetermined number of iterations without improvement of the best found solution [675].
- Achieving a predetermined number of times a percentage of neighbors at each temperature is accepted; that is, a counter increases by 1 each time a temperature is completed with the less percentage of accepted moves than a predetermined limit and is reset to 0 when a new best solution is found. If the counter reaches a predetermined limit R, the SA algorithm is stopped [420].

SA compared to local search is still simple and easy to implement. It gives good results for a wide spectrum of optimization problems: the historical ones such as TSP and VLSI design in different domains of application. A good survey on SA can be found in Refs [1,489,733].

2.4.3 Other Similar Methods

Other similar methods of simulated annealing have been proposed in the literature, such as threshold accepting, great deluge algorithm, record-to-record travel, and demon algorithms (Fig. 2.26). The main objective in the design of those SA-inspired algorithms is to speed up the search of the SA algorithm without sacrificing the quality of solutions.

2.4.3.1 Threshold Accepting Threshold accepting may be viewed as the deterministic variant of simulated annealing [228]. TA escapes from local optima by accepting solutions that are not worse than the current solution by more than a given threshold Q. A deterministic acceptance function is defined as follows:

$$P_i(\Delta(s, s')) = \begin{cases} 1 & \text{if } Q_i \geq \Delta(s, s') \\ 0 & \text{otherwise} \end{cases}$$