RAIRO. RECHERCHE OPÉRATIONNELLE

D. DELAMARRE
B. VIROT

Simulated annealing algorithm : technical improvements

RAIRO. Recherche opérationnelle, tome 32, n° 1 (1998), p. 43-73
<http://www.numdam.org/item?id=RO_1998__32_1_43_0>

Recherche opérationnelle/Operations Research
(vol. 32, n° 1, 1998, pp. 43-73)

SIMULATED ANNEALING ALGORITHM: TECHNICAL IMPROVEMENTS (*) (1)

by D. DELAMARRE and B. VIROT (2)

Communicated by Catherine ROUCAIROL

Abstract. – *We present an overview of the main* problem-independent *sequential and parallel amelioration techniques of the Simulated Annealing algorithm.*

We begin by briefly exposing the theoretical framework encompassing the standard markovian model, the notion of cycle and the optimal temperature schedules. Theory of cycles yields explicit relationships between the geometry of the energy landscape and the expected behavior of the algorithm. It leads to the design of efficient temperature schedules, as well as to improvements of the algorithm behavior by distorting the cost function.

Next, we present a survey of parallelization techniques, focussing on problem-independent synchronous strategies. They provide flexible and general tools, allowing operational research practitioners to take advantage of the computational power of parallel architectures.

We conclude with an application. It concerns the search for Hamiltonian paths in cubic graphs. It brings to the fore the efficiency of the cost function distortions technique, when used in combination with Parallel Simulated Annealing. © Elsevier, Paris

Keywords: Simulated Annealing, Heuristics, Optimization, Temperature Schedule, Cost Function Distortion, Parallelization, Cubic Graph, Hamiltonian Path.

Résumé. – *On présente une synthèse des principales techniques d'amélioration de l'algorithme du recuit simulé. On se limite aux techniques génériques, séquentielles ou parallèles, indépendantes du problème traité.*

On commence par exposer brièvement le cadre théorique, comprenant le modèle markovien classique, la notion de cycle et les schémas de température optimaux. La théorie des cycles permet d'obtenir des relations explicites entre la géométrie du paysage d'énergie et la vitesse de convergence de l'algorithme. Elle permet d'élaborer des schémas de température efficaces. Elle conduit à la méthode des distorsions de la fonction de coût, améliorant significativement la qualité des solutions pour certains problèmes difficiles.

Ensuite, on discute les techniques de parallélisation de l'algorithme du recuit simulé, en mettant l'accent sur les méthodes synchrones indépendantes du problème traité. Elles fournissent des outils génériques permettant aux praticiens de la Recherche Opérationnelle de tirer aisément parti de la puissance des architectures parallèles.

(*) Received June 1995.

(1) This work has been partly supported by the French CNRS Coordinated Research Programs "Coopération, Communication et Concurrence" and "Parallélisme, Réseaux et Systèmes" (projet LOREST, opérations CAPA et STRATAGEME).

(2) Laboratoire d'Informatique Fondamentale d'Orléans, Université d'Orléans, B.P. 6759, 45067 Orléans cedex 2, France. (E-mail : Bernard.Virot@lifo.univ-orleans.fr)

Recherche opérationnelle/Operations Research, 0399-0559/98/01/

On termine par une application. Elle concerne le problème de la recherche d'un chemin hamiltonien dans un graphe cubique. Elle permet de mettre en évidence l'efficacité de la méthode des distorsions de la fonction de coût utilisée en conjonction avec l'algorithme du recuit simulé.

Mots clés : Recuit simulé, Heuristique, Optimisation, Schéma de température, Distorsion, Fonction de coût, Parallélisation, Graphe cubique, Chemin hamiltonien.

CONTENTS

1. INTRODUCTION

The Simulated Annealing algorithm was introduced, as an optimization method, in 1982 by Kirkpatrick, Gelatt et Vecchi [27]. From mathematical point of view, this stochastic algorithm allows the minimization of a numeric

function (cost function or energy) $x \rightarrow E(x)$, where the variable x represents the current state of the system. In usual combinatorial applications, the space ε of all the possible states is very large and the function E has numerous local minima. Therefore iterative improvement algorithms – such as the steepest descent method – are inoperative. Similar to the methods by iterative improvement, the simulated annealing algorithm produces a sequence of approximate solutions. However, unlike the preceding methods, it can generate moves which increase the cost $E(x)$. These are accepted according to a law of probability suitably chosen, controlled by a parameter called temperature, which permits to escape from a local minimum. To obtain solutions close to the optimum, one classically decreases the temperature according to an appropriate law, defining in this fashion the *temperature schedule.*

The Simulated Annealing algorithm has been successfully applied to various combinatorial problems in the fields of operational research, image processing, computer assisted design of VLSI circuits, artificial intelligence... However, numerous experimentations show the difficulty to obtain a solution of good quality in a short time frame. Many researchers have shown interest to the problem of improving the Simulated annealing algorithm. The existing techniques in the literature can be classed in two categories.

The first technique concerns the fine tuning of the parameters of the algorithm, thereby improving the computation time and the quality of the solutions. Several approaches exist based on the temperature schedule, the cost function, the generation of the moves or the acceptance probability law. It is possible to found the choice and the adaptation of the temperature schedule by using an analysis of the distribution of energy like White [33], Otten and van Ginneken [28]. A second possibility of research lies in the adaptation of the generation of the moves. White [33] defines the concept of scale of moves. He proposes to favor the generation of large scale moves at high temperature and smaller ones at low temperature. Ingber ([24], [25]) proposes a choice of functions of generation in the particular case of product spaces. Some researchers have proposed to modify the acceptance probability law to improve the speed of convergence ([29], [16]). However, it is a difficult task to give general *a priori* estimations of the effect of these heuristics on the expected behavior of the algorithm.

The second technique consists in the designing of parallel algorithms, intended to take advantage of the computational power of supercomputers. The problem of parallelization of the Simulated Annealing algorithm has been the object of numerous studies, showing difficulty to obtain significant speed-

ups [20]. The constraints imposed by the mathematical model, which result in a strong coupling between processes, impose numerous synchronizations, therefore are expensive in terms of time. *Asynchronous* algorithms reduce the synchronizations by accepting to calculate changes of the cost function based on non up-to-date data ([20], [21]). In the existing state of art, it seems difficult to correctly control the admissible error, except for some specific problems ([14], [15], [21]). In contrast with this approach, *synchronous* algorithms guarantee the exact computation of the values of the cost function by using updated data. Problem-independent synchronous algorithms rely on simultaneous and coherent explorations of the space of states by the processors. They provide flexible and general tools, allowing operational research practitioners to safely use parallel architectures. These coarse grain parallelizations can be completed by problem-dependent data or task partitioning techniques.

As noticed by Eglese [16], in order to design and optimize such techniques, a crucial point is to have some theoretical results on the average or expected performance of the algorithm. A first attempt in this direction was performed by Hajek and Sasaki [23] for the Maximum Matching Problem. Recently, a more general approach was proposed by Azencott, Catoni and Trouvé [6]. The theoretical framework is furnished by the notion of *cycle*, as introduced by Friedlin and Wentzell in the context of dynamic systems [17]. Theory of cycles yields explicit relationships between the geometry of the energy landscape and the expected behavior of the algorithm, thereby allowing safe design of amelioration techniques, suitable for sequential algorithms as well as for parallel ones.

In this paper, our aim is to present an overwiew of the main *problem-independent* sequential and parallel amelioration techniques of the Simulated Annealing algorithm. We begin by briefly exposing the theoretical framework encompassing the standard Markovian model, the notion of cycle and the optimal temperature schedules. We show that it is possible to improve the algorithm behavior by distorting the cost function. Next, we discuss parallel synchronous algorithms, focusing on problem-independent ones. We conclude with an application. It concerns the search for Hamiltonian paths in cubic graphs. It brings to the fore the efficiency of the method of cost function distortions, when used in combination with Parallel Simulated Annealing.

2. MATHEMATICAL MODEL

2.1. The Markovian model

We define E as the energy function to be minimized, which is defined on

the set $\mathcal{E}$ of the states of a system. We define a family of partial mappings of $\mathcal{E}$ into itself, called moves, or elementary transformations. A state $y \in \mathcal{E}$ is a neighbor of a state x if there exists an elementary transformation from x to y. We define a transition matrix $Q = (q_{xy})$ on $\mathcal{E} \times \mathcal{E}$ such that

$$q_{xy} > 0 \quad \text{iff } y \text{ is neighbor of } x \text{ and } y \neq x$$

The Simulated Annealing algorithm can be modelled as the evolution of a Markov chain (X_n) controlled by a sequence (T_n) of parameters, called temperatures. Suppose $X_0, \ldots, X_n$ are built. We then choose at random an elementary transformation of the state X_n into Y_n according to the law

$$P(Y_n = y | X_0, \ldots, X_n) = q_{X_n y},$$

then, we choose at random X_{n+1} among Y_n and X_n, according to the law

$$P(X_{n+1} = Y_n | X_0, X_1, \ldots X_n, Y_n) = \min\left(1, \exp\left(\frac{-\Delta E}{T_n}\right)\right),$$

where ΔE represents the variation of energy corresponding to the elementary transformation of X_n into Y_n. The sequence (T_n) is called *temperature schedule*. If $X_{n+1} = Y_n$, one says the chosen move is *accepted* or that we have made a *step*, else $(X_{n+1} = X_n)$ one says the move is *rejected*. Note that the subscript n represents the time, namely the number of trials effectuated so far. A *chain* is a finite sequence $c = (s_i)_{1 \leq i \leq p}$ of states such that, for every i, s_{i+1} is a neighbor of s_i. If, moreover, $s_1 = x$ and $s_p = y$, one says that the chain joins x to y. The number $h(c) = \sup_{1 \leq i \leq n} E(s_i)$ is called the *height* of the chain.

We will suppose that the transition matrix Q is *symmetrical* and *irreducible*: for any pair of states x, $y \in \mathcal{E}$, $q_{xy} = q_{yx}$ and, moreover, there always exists a chain which joins x to y.

2.2. The notion of cycle in Simulated Annealing

The notion of cycle was introduced in the context of dynamic systems by Friedlin and Wentzell [17], and in the framework of Simulated Annealing by Azencott [6]. It furnishes a convenient framework to bind the convergence properties of the algorithm to a small number of simple parameters which describe the geometry of the energy landscape. The approach consists in the study of the probability of presence of the current state in a given subset of

the state space. Amongst these subsets, Friedlin and Wentzell consider the family of *cycles*, determined solely by the cost function and the elementary transformations. Intuitively, a cycle is a maximal set of states which can be joined by chains whose heights are less or equal to a given cost h. The following definitions formalize the notion of cycle.

DEFINITION 1: *Two states x and y are equivalent modulo h, if $x = y$ or if there exists a chain of height less or equal to h joining x to y.*

As the transition matrix Q is symmetrical, equivalence modulo h defines an equivalence relation $\mathcal{C}_h$ over $\mathcal{E}$. The equivalence classes are called cycles of level h. We denote by Γ_h the set of cycles of level h, and by $\Gamma = \bigcup_{h>0} \Gamma_h$ the set of all the cycles. Note that a singleton $\{x\}$ is a cycle of level h for any $h < E(x)$.

Remark: If $h < h'$ then $\mathcal{C}_h$ refines $\mathcal{C}'_h$. Therefore, two cycles are either disjointed or included one in another. The decomposition of the space of states into cycles can be represented by a tree which root is the state space $\mathcal{E}$ as a whole, and the leaves its singletons (see *Fig.* 1). The border of a cycle γ is the set $B(\gamma)$ of the states $y \in \mathcal{E} - \gamma$ which are neighbor of a state of γ. Note that, if $\gamma \in \Gamma_h$ is not a singleton, then for every $x \in \gamma$, $E(x) \leq h$ and for every $y \in B(\gamma)$, $E(y) > h$. The next definition summarizes the main geometrical characteristics of a cycle.

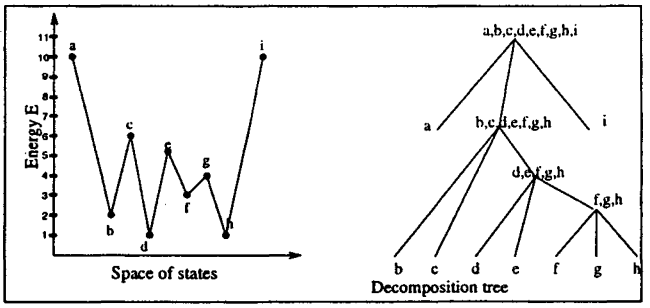


Figure 1. – In this figure we display, on the left, the energy landscape (two neighboring states are linked by a line), and on the right, the decomposition tree into cycles.

DEFINITION 2: *Let $E_{\min} = \inf_{x\in\mathcal{E}} E(x)$. For every cycle γ we define:*
the height of exit $H_e(\gamma) = \inf_{y\in B(\gamma)} \sup_{x\in\gamma}(E(y) - (E(x))^+$,
the potential $U(\gamma) = \inf_{x\in\gamma} E(x) - E_{\min}$,
the difficulty $D(\gamma) = \frac{H_e(\gamma)}{U(\gamma)}$.

2.3. Speed of convergence

Let us denote by $\mathcal{E}_{\min} = \{x \in \mathcal{E} | E(x) = E_{\min}\}$ the set of optimal states of $\mathcal{E}$. To estimate the performance of the algorithm, Catoni [7] evaluates, in term of time n, the decrease of the probability $P(X_n \in \mathcal{E}_{\min})$, called speed of convergence (according to Hwang-Sheu).

2.3.1. *Asymptotically good temperature schedules*

A temperature schedule is *asymptotically good* if it guarantees, with probability 1, the convergence to an optimal state, namely if $\lim_{n\to\infty} P(X_n \in \mathcal{E}_{\min}) = 1$. In his Thesis, Catoni established the existence of an explicit upper bound for the speed of convergence of asymptotically good temperature schedules [7]. Let us denote by $D = \max\{D(\gamma) | \gamma \in \Gamma\}$ the difficulty of the energy landscape. Catoni shows that, for every asymptotically good temperature schedule,

$$P(X_n \notin \mathcal{E}_{\min}) \geq \left(\frac{K}{n}\right)^{\alpha_{\text{opt}}},$$

where $\alpha_{\text{opt}} = \frac{1}{D}$ is a constant which only depends on the energy landscape, while K also depends on the temperature schedule. The exponent α_{opt} is called the *optimal speed exponent.*

A temperature schedule is *optimal* if it achieves the best speed of convergence (for some constant K), namely if:

$$P(X_n \notin \mathcal{E}_{\min}) \sim \left(\frac{K}{n}\right)^{\alpha_{\text{opt}}}$$

If one completely knows the structure of the energy landscape, then it is possible to design optimal schedules, by using triangular methods [8]. Otherwise, one has to use less accurate schedules, which are not optimal in general. We turn now to the simplest ones, namely logarithmic and exponential schedules.

2.3.2. *Logarithmic temperature schedules*

A temperature schedule (T_n) is *logarithmic* if the temperature is in the form

$$T_n = \frac{C}{\ln n}, \tag{1}$$

where C is some constant. Hajek [22] has proven that there exists a smallest constant C_{opt}, such that, for all $C \geq C_{\text{opt}}$, logarithmic schedules are asymptotically good. Catoni has shown that the value of C_{opt} is closely

related to the difficulties of cycles [7]. For asymptotically good logarithmic schedules:

$$P(X_n \notin \mathcal{E}_{\min}) \sim \left(\frac{K}{n}\right)^{\alpha},$$

where $K > 0$ and $\alpha > 0$ are appropriate constants dependent on the structure of the energy landscape and on the temperature schedule ([6], [9]). In the general case α is strictly greater than the optimal speed exponent α_{opt}. Thus, logarithmic schedules are not optimal schedules. Practically, it is a difficult task to estimate C_{opt}. Moreover, α is very small (we can even make it arbitrarily small by considering specifically created energy landscapes). Therefore, logarithmic schedules are not of great use, even though they are asymptotically good schedules.

2.3.3. *Exponential temperature schedules*

An exponential temperature schedule is a schedule (T_n) such that $T_n = a^n T_0$, where a is a constant, $0 < a < 1$ and a very close to 1. In contrast with logarithmic schedules, exponential schedules are not asymptotically good schedules. However, Catoni [8] has shown that they have good robustness properties, as long as one considers the solution obtained within a fixed number of trials (Time Bounded Annealing).

Fix a number N of trials to be performed and choose $T_n = a_N^n T_0$, where T_0 does not depend on N and $a_N = (c \ln N)^{\frac{1}{N}}$. Then, one guarantees the property

$$P(X_n \notin \mathcal{E}_{\min}) \sim \left(\frac{K'}{N}\right)^{\alpha},$$

where α is the largest exponent that one can obtain with a logarithmic schedule. The constants K', T_0 and c are simple to express in terms of cycles. Exponential schedules are not optimal in general. However, they are, in some sense, almost optimal [8]. Moreover, it is possible to show that the speed of convergence of an exponential schedule is less affected by a poor choice of the parameters than the speed of convergence of a logarithmic schedule [8]. Thus, exponential schedules are to be prefered to logarithmic ones.

Practically, one often adopt a temperature schedule running by stages. For such a schedule, we decompose the series of trials in consecutive stages $p_1, \ldots, p_k, \ldots$, of lengths $L_1, \ldots, L_k, \ldots$, respectively. On each stage, one choose a constant temperature Θ_k. It decreases from stage according to

the equation $\Theta_k = a^k \Theta_0$, where the constant $a < 1$ is chosen close to 1. If the length of the stages increases exponentially with k, $(L_k = l^k L_0)$, such a schedule is an approximation of an exponential schedule ([2], [4]). Therefore, we can reuse the previous discussion in this framework as well.

3. IMPROVEMENTS OF SIMULATED ANNEALING

3.1. Repeated independent annealings

As pointed out by Azencott [6], in the framework of Time Bounded Annealing it can be preferable to perform several independent annealings of shorter (time) length rather than only one of a given length. For instance, suppose that one disposes of a fixed amount of time allowing N trials, and that the temperature schedule gives a speed of convergence of the form

$$P(X_n \notin \mathcal{E}_{\min}) \sim \left(\frac{K}{N}\right)^{\alpha}.$$

Then, instead of realizing a single annealing of lenght N, we can perform s successive independent annealings with the same temperature schedule. We stop each execution after $n = \frac{N}{s}$ trials. Denote by $\{X_{n,1}, \ldots, X_{n,s}\}$ the s final states. We then choose a state $Y_n \in \{X_{n,1}, \ldots, X_{n,s}\}$ such that $E(Y_n) = \min\{E(X_{n,1}), \ldots, E(X_{n,s})\}$. We obtain

$$P(Y_n \notin \mathcal{E}_{\min}) = \prod_{1 \leq i \leq s} P(X_{n,i} \notin \mathcal{E}_{\min}) \sim \left(\frac{Ks}{N}\right)^{\alpha s}.$$

By using s successive annealings, it is therefore possible to replace the exponent α by αs at the cost of an increase of the constant K.

From the values of K and α one can easily compute the number s of successive annealings that optimizes the speed of convergence. A *Multiple Optimal Simulated Annealing* consists in the repetition of $s_N \sim N/eK$ successive independent annealings of fixed length $n \sim eK$ (with e the basis of the exponential) and in the choice of the best final state Y_n. We then obtain an optimal speed of convergence of the form

$$P(Y_n \notin \mathcal{E}_{\min}) \sim e^{-\rho N}, \quad \text{with } \rho = \alpha/2eK$$

Remark: As $e^{-\rho N}$ is optimal, one sees that performing repeated independent annealings can increase the speed of convergence if and only if

$e^{-\rho N} < \left(\frac{K}{N}\right)^{\alpha}$. This condition boils down to

$$\frac{K}{N} \log \frac{N}{K} < \frac{1}{2e}.$$

Note that the last inequality does not depend on α.

3.2. Distortions of the cost function

To improve the speed of convergence, one can increase the optimal speed exponent $\alpha_{\text{opt}} = \frac{1}{D}$, by decreasing the difficulty D of the energy landscape. With this aim in view, Azencott [5] proposes to replace the cost function E by $f(E)$ where the function f is non decreasing and strictly concave. Intuitively, this replacement preserves the set $\mathcal{E}_{\min}$ of optimal states and it yields an energy landscape with less deep cycles, thereby decreasing its difficulty D. In fact, suppose we use an optimal temperature schedule such that

$$P(X_n \notin \mathcal{E}_{\min}) \sim \left(\frac{K}{n}\right)^{\alpha_{\text{opt}}},$$

where α_{opt} is the optimal speed exponent attached to the energy function E. Let us replace E by $f(E)$, where the function f is non decreasing, strictly concave and continuously differentiable. Denote by α^f_{opt} the new optimal speed exponent attached to $f(E)$. By the definition of difficulty D, there exists a cycle γ and two states $u \in B(\gamma)$ and $v \in \gamma$ such that

$$\alpha^f_{\text{opt}} = \frac{\inf_{x\in\gamma} f(E(x)) - \inf_{x\in\mathcal{E}} f(E(x))}{f(E(u)) - f(E(v))}$$

As f is non decreasing, we have

$$\inf_{x\in\gamma} f(E(x)) = f(\inf_{x\in\gamma} E(x)) \qquad \text{and} \qquad \inf_{x\in\mathcal{E}} f(E(x)) = f(E_{\min})$$

Let us choose two states $a \in \gamma$ and $b \in \mathcal{E}$ such that $E(a) = \inf_{x\in\gamma} E(x)$ and $E(b) = E_{\min}$. We have $E(u) > E(v) \geq E(a) > E(b)$. As f is strictly concave, a straightforward computation yields

$$\alpha^f_{\text{opt}} = \frac{f(E(a)) - f(E(b))}{f(E(u)) - f(E(v))} > \frac{E(a) - E(b)}{E(u) - E(v)} \geq \alpha_{\text{opt}}$$

As $\alpha^f_{\text{opt}} > \alpha_{\text{opt}}$, by using an optimal temperature schedule adapted to the cost function $f(E)$, we obtain a speed of convergence strictly improved. Azencott suggests, for example, the use of distorsion functions f as

▷ $f(x) = -e^{-bx}$ with $b > 0$,

▷ $f(x) = (x + a)^{1/r}$ with a greater so that $E(s) + a \geq 0$ for every state s,

▷ $f(x) = \ln(x + a)$, for the same values of a.

In the last section, we will show that it is possible to compute "adequate" distortion functions. Experiments show that they yield significant improvements of the algorithm behavior, even when used with non-optimal schedules such as classical exponential schedules.

4. PARALLELIZATIONS OF SIMULATED ANNEALING

The problem of parallelizing the Simulated Annealing algorithm has recently been studied in numerous experiments, showing the difficulty to obtain significant speed-ups [20]. The constraints imposed by the underlying mathematical model are illustrated by a strong coupling between the processes, imposing numerous synchronizations which are expensive in terms of time. The existing approaches can be subdivided into two sub-categories [20].

▷ The *asynchronous (inexact)* algorithms reduce the amount of synchronizations by accepting to compute the variations of the cost function from non up-to-date data. In the current state of art, it seems difficult to correctly control the acceptable error, except for some specific problems [14];

▷ The *synchronous (exact)* algorithms, on the contrary, guarantee an exact computation of the cost function by using coherent data.

As the behavior of asynchronous algorithms can hardly be predicted, excepted for some specific problems ([14], [15]), we will focus on synchronous algorithms.

4.1. Principles of synchronous algorithms

One can imagine to parallelize the internal functions of a problem, by decomposing the management of the problem-dependent data into parallel tasks (task parallelism). This kind of parallelization is limited by the parallelism that one can extract from the move generation and evaluation functions. It can be successfully applied to large irregular data structures, such as graphs. One can also think of distributing among the processors the data describing the problem (data parallelism). Each processor then computes

moves modifying its own variables. This approach leads to medium or fine grain parallelizations. It is especially interesting when the treated problem involves large regular data structures (image processing for instance).

However, these parallelization techniques are strongly dependent on the treated problem. They do not preserve the genericity of the sequential Simulated Annealing algorithm. In contrast with these *specific* approaches, there exists parallelizations techniques that allow to design and implement *generic* parallel algorithms. As they are not bound to a particular class of problems, the same parallel harness can be reused for different applications. As all of these algorithms rely on simultaneous evaluations of several series of moves by the processors, it is possible to design libraires of parallel functions, thereby facilitating implementations and experimentations of multiple strategies on different parallel architectures. This approach was successfully developed by Delamarre [11].

We turn now to a brief overview of the synchronous problem-independent parallel algorithms.

4.2. Independent multiple annealings

As we have previously noticed, it can be more interesting, in the sequential case, when one disposes of a bounded time, to effectuate several short independent annealings, with the same temperature schedule, rather than a long one (*cf.* section 3.1). These independent shor annealings can obviously be performed in parallel. Then, Azencott [6] shows that one can reach the same solution quality as in sequential computation in a shorter time. However, the speedup is poor. Moreover, these parallelizations are hardly extensible since each of the independent annealings must conserve a sufficient length to insure its convergence.

Remark: A possible improvement consists in using repeated Simulated Annealings on each processor, with the same *optimal* temperature schedule. Azencott points out that this strategy would yield a linear speedup of order p with p processors [6]. However, the method is merely of theoretical interest, as it supposes that one is able to design an optimal temperature schedule.

4.3. Multi-temperature annealings

Another approach consists in the distribution of the sequence of temperatures on the processors. Thus, each processor executes an annealing with a fixed temperature. The processors synchronize and interact perio-

dically. This class of parallelizations of simulated annealing is called *multi-temperature* annealings. Two variants are to be distinguished.

▷ *Deterministic propagation of states*: When synchronizing, one propagates in a deterministic way the best states from high temperatures towards the lower ones [19].

▷ *Probabilistic exchange of states*: When synchronizing, one exchanges the obtained states according to a probabilistic criterion aimed at approaching a thermic quasi-equilibrium [26].

4.3.1. *Deterministic propagation of states*

The interaction scheme is described hereafter (*Fig.* 2). Each processor π_k holds a constant temperature T_k, $(T_k > T_{k+1})$. Processors synchronize every s iterations. when synchronizing, π_k holds a final state X_k and the processors execute the following algorithm

$$\begin{aligned} &\text{for all } 1 \leq k < p \\ &\quad \text{if } E(X_k) < E(X_{k+1}) \text{ then} \\ &\quad\quad X_{k+1} := X_k; \end{aligned}$$

In this way, better states are propagated towards lower temperatures.

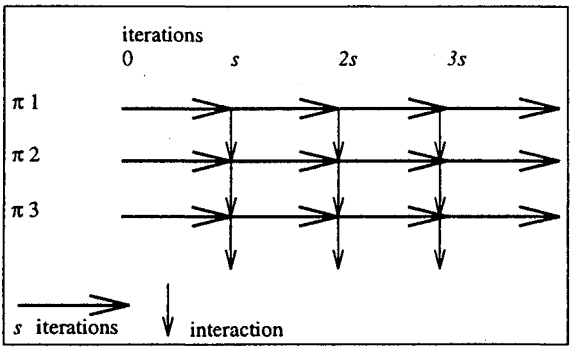


Figure 2. – Interactions in deterministic propagation of states.

Using this strategy, Graffigne [19] has performed experiments showing that it is possible to obtain good convergence results for difficult problems.

A variant. At high temperature, almost all the moves are accepted. Each accepted move costs – in addition to the time used to generate, evaluate and

decide to accept or reject it – the time of its realization. Therefore, a processor with a high temperature will be longer, in average, to perform s iterations than a processor with a lower temperature. If we synchronize all processors every s iterations, we make the processors using a low temperature wait for the processors using the highest ones.

To overcome this difficulty, we do not stop the processors using a low temperature as soon as they have computed s iterations. We rather let them going on computing moves until they are stopped by their higher temperature neighbor. When the processor π_1 which uses the highest temperature has done s iterations, then it synchronizes with its immediate lower temperature neighbor. The subsequent processors synchronize by pairs using the following algorithm, thus propagating a synchronization wave towards the lowest temperatures.

```
if k > 1 then
        synchronize (π_{k-1});
        if E(X_{k-1}) < E(X_k) then
            X_k := X_{k-1};
if k < p then
        synchronize (π_{k+1});
```

The function synchronize (π_k) realizes a *rendez-vous* between the processor which executes it and the processor π_k. This function terminates when the two processors have executed the corresponding calls. Using this variant, we can take advantage of some overlapping between communications and computations. We prevent processors which hold low temperatures from being idle while waiting for the ones owning high temperatures.

We have made a group of experiments conducted with this variant for two classical problems, the Quadratic Assignment Problem (20 to 80 locations) and the Traveling Salesman Problem (50 to 500 towns). We used a network of 13 workstations. The main difficulty was to do a "good" choice of temperatures. We tested two simple strategies. Let us denote by $T_1 = T_\infty$ the (highest) temperature on processor π_1 and by $T_p = T_{\text{low}}$ the lowest one on processor π_p. We have choosen T_∞ in a simple way, as suggested in [2] for simple temperature schedules. We have perform experiments with two different strategies for the distribution of intermediary temperatures.

▷ an arithmetical progression: $T_k = T_\infty - (k-1) * \frac{T_\infty - T_{\text{low}}}{p}$,

▷ a geometrical progression: $T_k = T_\infty * (T_\infty / T_{\text{low}})^{\frac{(k-1)}{p}}$.

The experimental results highlight a better speedup with a geometrical progression. Moreover, they showed that T_{low} must be low enough (100 to 1000 times less than T_∞). Otherwiwe, the mean energy at this temperature would be strictly greater than the optimum, leading to a poor convergence. One would therefore have to store explicitly the best solution encountered so far.

Experiments conducted with the variant showed an increase in speedup of the order of 10% to 20% (depending on the choice of T_{low}), when compared with the algorithm using global synchronizations.

4.3.2. *Probabilistic exchange of states*

This strategy of multi-temperature annealing was proposed by Kouichi Kimura and Kazuko Taki [26]. In contrast with the deterministic approach, every s. iterations one interchanges the states between processors with neighboring temperatures, using a suitable probability of exchange p_e. The probability of exchange is determined in a way to promote the thermic quasi-equilibrium. Let us denote by $Z(T) = \sum_{x \in \mathcal{E}} \exp\left(\frac{E(x)}{T}\right)$ the Boltzmann partition function. If $(T_k - T_{k+1})(E(X_k) - E(X_{k+1})) < 0$, as we obtain a lower energy for a lower temperature, then we choose $p_e = 1$. On the contrary, if $(T_k - T_{k+1})(E(X_k) - E(X_{k+1})) \geq 0$, then the following equation, expressing the conservation of the thermic equilibrium, should be verified:

$$\frac{1}{Z(T_k)} \exp\left(\frac{E(X_k)}{T_k}\right) \frac{1}{Z(T_{k+1})} \exp\left(\frac{E(X_{k+1})}{T_{k+1}}\right) = \frac{1}{Z(T_k)} \exp\left(\frac{E(X_{k+1})}{T_k}\right) \frac{1}{Z(T_{k+1})} \exp\left(\frac{E(X_k)}{T_{k+1}}\right) p_e$$

Therefore, we choose

$$p_e(T_k, E(X_k), T_{k+1}, E(X_{k+1})) = \begin{cases} 1 & \text{if } \Delta T \cdot \Delta E < 0, \\ \exp\left(-\frac{\Delta T \cdot \Delta E}{T_k T_{k+1}}\right) & \text{otherwise,} \end{cases}$$

where $\Delta T = T_k - T_{k+1}$ and $\Delta E = E(X_k) - E(X_{k+1})$.

It has been shown by Kouichi Kimura and Kazuko Taki that, if one uses the previous probability of exchange p_e then, for each processor, the

distribution of states approaches in a monotonous manner, in the sense of Kullback's distance, the equilibrium distribution [26].

Kouichi Kimura and Kazuko Taki have performed experiments for a problem of graph separation. They pointed out an improvement of the solutions quality with respect to the sequential case for the same time of computation.

An exchange of states between two processors π_k and π_{k+1}, having the temperatures T_k and T_{k+1} respectively, amounts to an exchange of the temperatures of the processors π_k and π_{k+1}. Therefore, the algorithm handles p Markov chains which change temperature by changing processors. During a synchronization phase, each chain "chooses" a temperature schedule by a series of tournaments with the chains at neighboring temperatures. Note that, according to the choice of p_e, these temperature schedules do not necessarily decrease. The problem of convergence of the chains arises.

Remark: The Parallel Systolic Annealing algorithm proposed by Aarts and van Laarhoven ([2], [1]) is close to the Multi-temperature Annealing with probabilistic exchange of states. It differs in the initialisation stage and in the choice of the exchange probability. This probability is computed by using an approximation of the partition function of the Boltzmann distribution.

4.4. Parallelization by multiple trials

A simple strategy of parallelization consists in using a processor farm to evaluate in parallel moves generated by a master processor. However, parallelizations based on the processor farm paradigm can present a bottleneck phenomenon due to the uniqueness of the move generator. To remedy this difficulty, one can think of distributing the task of the move generation equally on the processors. This leads to the parallelization strategies by *multiple trials*. The p processors start from the same state. Then, each one independently computes a chain. The length of these chains is controlled by previously fixed parameters. Then, one synchronizes the processors and chooses the final state generated by one of the processors to restart the procedure. The speedup obtained in this case comes from the fact that at low temperature, the *acceptance rate* (*i.e.* the rate of accepted moves compared to the generated ones) is small. Then, one can hope to accept a given number of moves by computing p chains in parallel in an order of p less time than by computing one sequential chain. Several strategies belongs to this category. One distinguishes them by their way of controlling the length of the chains.

▷ In a first group, one places the strategies for which the generated chains correspond to a previously fixed number of *accepted* moves.

▷ In a second group, we place the strategies for which the generated chains correspond to a fixed number of *trials* ([4], "division algorithm").

Moreover, one can dynamically adjust the granularity of the parallelization, either by increasing the number of processors, or by modifying the way of grouping the processors during the annealing ([13], [2]).

We turn now to the description of these strategies, following the two criteria of the chain length counting and of the fixed or variable granularity.

4.4.1. *Fixed granularity, length of chains in number of trials*

One can think of controlling the chains length in terms of the number of trials. This leads to the division algorithm proposed by Aarts and Debont [1]. If we dispose of p processors, then we can replace the computation of a chain of length L by the construction of p chains of length L/p, called "sub-chains". Note that the length of these sub-chains is expressed by a number of trials. A difficulty arises, as this strategy leads to a hardly extensible algorithm. In fact, when p increases, the length of the sub-chains becomes too small to authorize a suitable convergence at low temperature.

4.4.2. *Fixed granularity, length of chains in number of accepted moves*

Instead of controlling the length of chains in terms of trials, one can use a fixed number of accepted moves as well.

Starting from the same initial state, the processors independently generate moves, until one of them, at least, has build a chain comporting a fixed number a of accepted moves. Then, one synchronizes all the processors and the final state of such a chain is chosen as the initial state for a new iteration. These methods have given rise to a number of experiments on many classical problems of discrete combinatorial optimisation, and are applicable on architectures having either a shared memory or a distributed one ([6], [13]).

Synchronization at the first accepted move. We first consider the simple strategy where the processors synchronize at the first accepted move ($a = 1$). We can show three variants of this strategy for which we give an *a priori* comparison.

▷ *Synchronous by trial strategy.* This parallelization strategy is also called "high temperature mode" by Roussel-Ragot and Dreyfus [30]. It makes use of the following algorithm to generate an accepted move with p processors.

For every processor π_k, $1 \leq k \leq p$
    generate, evaluate and test a move m_k;
global-synchro;
if there exists at least an accepted move then
    choose an accepted move m randomly among them;
    realize the move m;

The function global-synchro synchronizes the p processors. This strategy uses too much synchronizations at low temperature. In fact, in this case, the acceptance rate can be very low, and all of the processors are often synchronized even though no accepted move has been found. Therefore, this stretegy is generally used with temperature sufficiently high such that the probability to obtain at least one accepted move at each synchronization is high enough.

▷ *Synchronous by move stragegy.* This strategy avoid all useless synchronizations while no accepted move is found. It is also called "low temperature mode" or "asynchronous parallelization" by Roussel-Ragot and Dreyfus [30]. However, as it uses synchronizations to maintain coherent updated data, it is part of the category of synchronous algorithms. Every processor π_k, $1 \leq k \leq p$ executes the following algorithm.

Repeat
    generate, evaluate and test a move m_k
until $\exists i \in \{1, \ldots, p\}$ such that m_i is accepted;
global-synchro;
realize the move m_i;

Note that the fact one restarts with the first accepted move introduces a bias in the choice of the move to be realized at high temperature. Indeed, at high temperature, almost every trial is accepted. Therefore, one promotes the moves which have the shortest time of computation. Even if the problem comports only one kind of move, the moves which cause an increase in energy are more slowly tested than energy decreasing ones since the former require the computation of an exponential.

▷ *Unbiased synchronous by move stragegy.* We can solve the problem of bias by using the variant called "one chain" strategy by Lülling [13]. This variant differs from the preceding one in the fact that during the synchronization, we allow the processors to finish the computation of the current move and take into account the fact that they can find an accepted move during this last computation. Lülling noticed that the speedup obtained increases with the number of processors until a certain number. Beyond this

limit, if additional processors are used, the performances decrease due to the synchronization and communications costs.

Synchronization after $a > 1$ accepted moves. Another interesting way to alleviate the problem of bias while avoiding useless synchronizations consists in synchronizing only when a processor has accepted a number $a > 1$ of moves. Virot [31] proposed an analysis of this strategy. The choice of the parameter a is important in the determination of the expected speedup. Virot describes a model showing that, for every temperature, it is possible to compute an optimal number of processors, which increases when the acceptance rate decreases. Moreover, this optimal number depends on the underlying architecture through the time required to synchronize the processors.

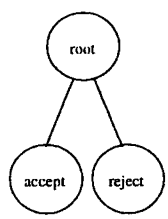
4.4.3. *Variable granularity*

Strategies with a fixed granularity bring to the fore the existence of an optimal number of processors, which increases when the temperature decreases. We could decide to use, with each temperature, only the optimal number of processors for this temperature, until the moment where all the processors are used. This approach is interesting in an environment of multi-computing where one can dynamically add processors (*i.e.* a network of workstations used as a multi-processor machine). The computation power unused by the annealing can serve to execute other programs. In the case of multi-processor architectures, where a set of processors is reserved at the beginning of the annealing computation, this practice would lead to a waste of computation power.

Two variants of a same solution were proposed by Arts and Debont [1] and Lülling [13]. Suppose the optimal number of processors n_{opt} is smaller than the number p of available processors. Then, one can compute each of the chains by using a parallel strategy on a *group* (or *cluster*) of processors. One performs, thus, a "multi-level" parallelization. Each group uses an Unbiased Synchronous by Move strategy for the computation of its chain. The manner in which the processors are grouped evolves dynamically along the decrease of the temperature. The difference between the two variants of Aarts-Debont and of Lülling holds principally in the way of grouping (sizes of the clusters and criterion to decide to group). The grouping criterion is, nevertheless, based on the same heuristics. It consists in doing so that the size of a group is kept as close as possible to n_{opt}.

4.5. Parallelization by speculative computation

The parallelization of Simulated Annealing by speculative computation makes use of the parallelism coming from the overlapping of the computation time of a move and of the next one [32]. We obtain this overlap by continuing the computation from a given state on two processors. These processors make a wager before the end of the computation of the current move. One processor bet that the given move will be accepted, the other that it will be rejected. In this way, one builds a binary tree which describes the successive possible choices: acceptance or rejection. On a binary tree of height one, the computation is performed in the following manner. The three processors start from the same initial state. Then



- ▷ the processor that bets on a rejection starts immediately the search for a move;
- ▷ the root processor generates a move m_k. It sends it to the processors wagging for its acceptance and goes on its computation for m_k;
- ▷ the processor betting on the acceptance (*accept son*) realizes the move m_k and generates its own move.

We formalize this behavior by the following algorithms on a height 1 tree:

for the root processor:

generate a move m_k;
send m_k to accept son;
evaluate and test move m_k;

for the others:

if accept son then
  receive (m_k);
  realize (m_k);
generate a move m_{k+1};
evaluate and test move m_{k+1};

When the root processor finishes its computation, then one knows which of the two wagerers holds the state corresponding to the exact computation of the Markov chain.

We show in Figure 3 the time schedule comparing the execution of the algorithm on a tree of height one and in sequence. We denote by

- ▷ "tm" the generation time of a move,
- ▷ "teval" the evaluation time,

▷ "tr" the realization time,

▷ "td" the time of test and decision,

▷ "tc" the communication time of a move.

Assume that computations and communications can overlap. Then, the time required to build a chain of length greater than 2 is clearly shorter with the parallel speculative computation ("tp") than in sequence ("ts"). This is even more true when the time of evaluation dominates the other times.

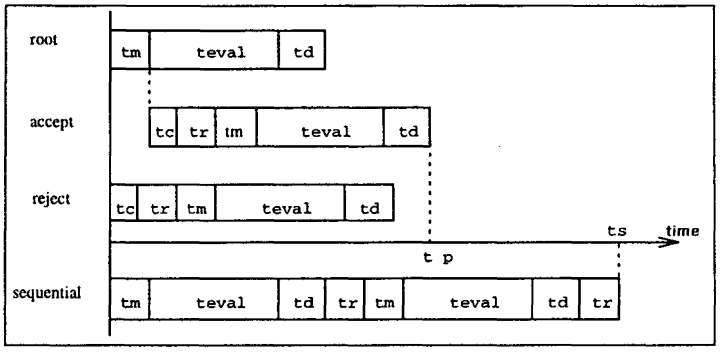


Figure 3. – Time schedule for a tree of height 1.

Clearly, one cannot dispose of a sufficient number of processors to develop a tree high enough to speculate on the results of all the iterations of an annealing. Consider a tree of bounded height. The effective successive decisions form a path from the root up to one of the leaves. Then, the processor situated at this leaf communicates its current state to the root processor, and the computation continues. The obtained speedup, in this case, is limited by the number h of processors having participated to the effective decisions. If we dispose of p processors and if we use a balanced binary tree, then we obtain a maximal speedup of $h = \log_2(p+1)$. Thus, the hypothesis of a balanced tree is unsuitable. One has to unbalance the tree according to the acceptance rate. The idea is to allow a path of effective decisions to descend the lowest possible in the tree before "exiting" by a leaf.

Witte, Chamberlain and Franklin [32] have proposed an analysis of the expected maximal speedup according to the shape of the binary tree, the acceptance rate and the times tm, teval and tc. For a given number p of processors, there exists an optimal tree, which shape depends on the acceptance rate and on the parameters tm, teval and tc. The construction of this tree requires an algorithm in time $O(p^4)$. Witte, Chamberlain and

Franklin propose an "approached" algorithm which yields a quasi-optimal tree in time $O(p)$. It needs a prediction of the acceptance rate for the next temperature stage. They applied the "approached" algorithm to a task-assignment problem. Using an Hypercube architecture the obtained speedup was 3.2 with 8 processors and 3.3 with 16 processors.

This static approach exhibits several inconveniences. We note that when the path of decision has passed a certain level in the tree, some subtress become useless (see *Fig.* 4). It would be beneficial, in a dynamic version to recycle these useless processors to prolong the useful subtrees. In addition, the processors situated at the top of the path should also be recycled. This poses the problem of choice of the place where a new leaf can be grafted and of the choice of the management of the migration of the processors in the tree.

Another problem arises from the necessity to estimate the times of computation and communication of a move in order to determine a suitable tree. This becomes particularly delicate on architectures such that a network of workstations used as a multi-processor machine. The load of the communication network is hardly predictable due to the sharing of the network with other users.

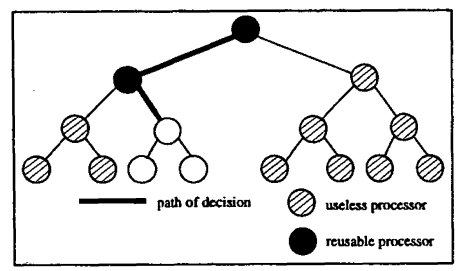


Figure 4. – Useless and reusable processors.

5. AN APPLICATION TO THE PROBLEM OF HAMILTONIAN PATH SEARCH IN CUBIC GRAPHS WITH COST FUNCTION DISTORTION

We turn now to an application. We choose a problem proposed by J. L. Fouquet and H. Thuillier [10]. It concerns the search for Hamiltonian paths in cubic graphs. It is a challenging problem, as classical approaches highlight the poor behavior of the Simulated Annealing algorithm when applied to difficult energy landscapes. In fact, the energy landscape presents deep cycles near local optima and, moreover, large plateaux yielding an important quantity of moves which leave the energy constant.

We proceed in two steps. A first modification of the cost function suppresses plateaux. Next, we use cost function distortions to get less deep cycles. Our experiments are conducted using a parallel Synchronous by Trials strategy (*cf.* 4.4.2), and a simple exponential temperature schedule.

5.1. Hamiltonian paths in cubic graphs

Given a non-directed graph G, we denote by $V(G)$ the set of its vertices and by $E(G)$ the set of its edges. We say that G is of order n if $|V(G)| = n$, and that it is *regular* if all of its vertices have the same degree k. We only consider finite graphs G regular, without cycles and of degree 3, called *cubic graphs*. If P is a path in G, we denote by $l(P)$ the length of P, namely the number of its edges. We extend this notation to all subgraphs H of G. We denote by $l(H)$ the length of the longest path in H. A path P is *Hamiltonian* if it passes exactly once by all the vertices of G. A forest is *linear* if all of its connected components are paths. Two forests F_1 and F_2 are disjoint if they do not have any common edge ($E(F1) \cap E(F2) = \emptyset$). If, in addition, $E(G) = E(F_1) \cup E(F_2)$, then we say that $F = (F_1, F_2)$ constitutes a linear partition of G. The number $l(F) = \max(l(F_1), l(F_2))$ is called the height of the linear partition.

Here, we call *Hamiltonian Problem*, the problem to decide if a cubic graph possesses a Hamiltonian path. The Hamiltonian Problem is NP-hard [18]. Let k be a fixed integer, $2 \leq k \leq n-1$. We show in [10] that the problem which consists in finding a linear partition of height k of G is a relaxation of the Hamiltonian Problem for this graph. In fact:

▷ if the graph G has a partition of height $n-1$, then it has a Hamiltonian path;

▷ conversely, if G has a Hamiltonian path P, by taking $F_1 = E(P)$ and $F_2 = E(G) \backslash E(P)$, we define a linear partition of height $n-1$.

Thus, to solve the Hamiltonian Problem for a cubic graph amounts to build a linear partition F with maximal height: $l(F) = n-1$.

Akayama, Exoo and Harary [3] have proved the existence of a linear partition for every cubic graph. One can find an algorithmic proof of this theorem in [10]. This last proof gives us a way to construct a linear partition. Moreover, we can go from a linear partition F to another by a simple transformation (*cf.* [10], section 3). Therefore, it is possible to use the Simulated Annealing algorithm in order to search for a linear partition with *maximal* height.

5.2. The algorithm

5.2.1. *The initial partition*

We use the algorithm exposed in [10] (section 2) to build the initial partition. This algorithm consists in an edge coloring with two colors. The coloring is done in a manner that the subgraph of each color constitutes a linear forest. Each time that an indifferent choice of colors arises, we choose a color at random.

5.2.2. *The moves generation*

The generation of the moves is described by the following algorithm (cf. *Fig.* 5).

choose at random $i \in \{1, 2\}$;
If there is no path with length $l > 2$ in F_i then
  stop the annealing {We have a Hamiltonian path}
else
  choose a non-terminal edge;
  change its color;

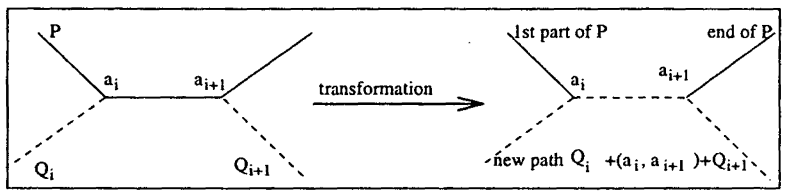


Figure 5. – A move.

5.2.3. *The cost function*

The easiest way to search for a linear partition of maximal length consists in the application of the Simulated Annealing algorithm to the maximization of the "sup" cost function $E_s(F) = l(F)$. This method leads to poor results. We can give at least two reasons for this bad behavior.

▷ There exists an important quantity of moves which keep E_s constant (*null* moves);

▷ when a forest has a very long path, breaking this path – except near its extremities – causes an important decrease of the cost function. Due to this fact, it is difficult to prevent the system from freezing (*i.e.* being trapped in a local optimum), even with an extremely slow decrease of temperature.

We therefore introduce another cost function. Let us denote by $\mu(F)$ the average length of the paths of a linear forest F, and by $\omega(F)$ the number of connected components in F. We define the "mean" cost function E by $E(F) = |\mu(F_1) - \mu(F_2)|$. The next proposition states that, as long as we search for Hamiltonian paths, we can safely replace E_s with E. It can be proved by classical graph-theory arguments (*cf.* [12]).

PROPOSITION: *if G possesses a Hamiltonian path, then E reaches its maximum value for exactly the states achieving the maximum of E_s.*

Remark: Note that this property is no longer true if the graph does not hold a Hamiltonian path. One can exhibit cubic graphs for which there exists a partition achieving the best value of the mean cost function and which does not yield the longest possible path. Therefore, this functions is well suited to the Hamiltonian path search but not to the longest path search in arbitrary cubic graphs.

As confirmed by experiments, the "mean" cost function E comports much less null moves than the "sup" cost function E_s, thereby avoiding the first cause of the bad behavior of E_s. Nevertheless, the function E still exhibits a pathological feature. As for the "sup" function, when a long path has been built, the next move cause a large variation of cost. One can only get out of such a local optimum by some moves causing a tremendous decrease of the cost or by a long sequence of smaller moves. Therefore, the system freezes easily when the temperature decreases.

5.3. Experimental results

Our experiments were conducted on a network of 13 Transputers, using a Synchronous by Trials strategy (*cf.* 4.4.2), and a simple exponential temperature schedule. We applied a classical end-of-stage criterion based on a minimal number of accepted moves while limiting the number of trials.

We used a generator of random Hamiltonian graphs or order n. For each size n, we perform 20 executions with randomly generated Hamiltonian graphs of order n. Let us call *success ratio* the rate of executions which yields a Hamiltonian path. We denote by N_{acc} the minimal number of accepted moves in each temperature stage. We maximize the "mean" cost function E. As shown by Table I, it does not give an optimal solution (a Hamiltonian path) except for small graphs. In Figure 6 we have plotted the acceptance rate and the mean value of $l(F)$ on each temperature stage for a

graph of order 100. The graphs highlight the brutal freezing of the system, due to the large size of the leaps of energy near local optima.

TABLE I
Statistics over 20 executions with randomly generated Hamiltonian graphs of order n: Mean Cost function.

$N_{acc} = \frac{n^2}{2}$	$n = 10$	$n = 50$	$n = 100$
sucess ratio	1	1	0.1
average final value of $l(F)$	9	49	70.1

$N_{acc} = \frac{n^2}{10}$	$n = 10$	$n = 50$	$n = 100$
sucess ratio	1	0.9	0
average final value of $l(F)$	9	48.7	63.0

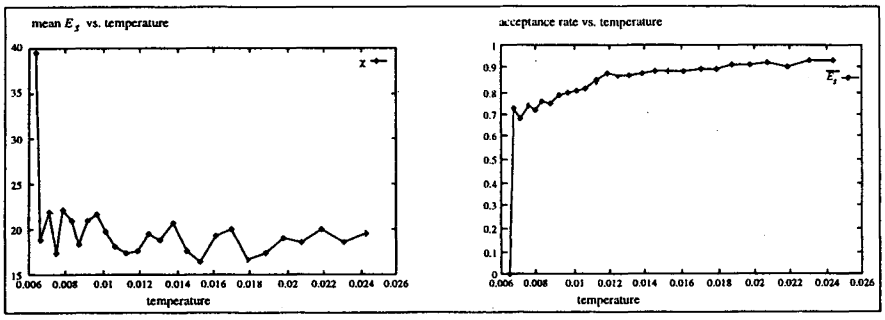


Figure 6. – Order 100 graph.

5.4. Cost function distortion

The existence of large energy leaps can be highlighted by a simple experiment. For any elementary move m bringing a state x to a state y, consider the *jump* $J(m) = |E(y) - E(x)|$. For each temperature stage, it is possible to observe the mean value $\overline{E}$ of the energy E and the mean value $\overline{J}$ of the jumps $J(m)$ corresponding to all of the tried moves. We have plotted the experimental values of $\overline{E}$ and $\overline{J}$ for a number of graphs of different sizes. The shape of the curve turns out to be very similar in all of the cases (*cf.* Figure 7 for a graph of size $n = 100$). It exhibits large values of $\overline{J}$ near the optimum of E, thereby bringing to the fore the pathological feature of the energy E.

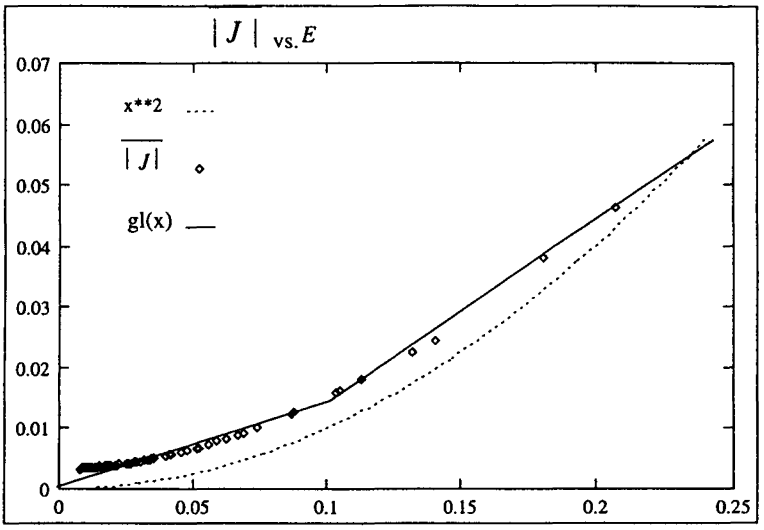


Figure 7. – $\overline{J}$, g_1 **and** g_2 **vs.** $\overline{E}$.

To overcome this difficulty, we use a simple heuristics based on the method of cost function distortion (*cf.* section 3.2). Consider a distortion $E' = f(E)$ of the cost function E, where the function f is non decreasing and strictly concave. Denote by $J'(m) = |E'(x) - E'(y)|$ the corresponding jumps. By considering a move as an elementary variation of the current state s, we have the approximation $J(m) \simeq \left| \frac{dE}{ds} ds \right|$, hence, we obtain

$$J' \simeq \left| \frac{df(E)}{ds} ds \right| = \left| \frac{df(E)}{dE} \frac{dE}{ds} ds \right| \simeq \frac{df(E)}{dE} J$$

To avoid large jumps near the optimum of $f(E)$, we can define f in such a way as J' stays approximately constant. Therefore, we can choose $\frac{df(E)}{dE} = \frac{1}{J}$. In order to compute f, we have to express J as a function of E and, then, to define f as a primitive of $\frac{1}{J}$. The guiding idea consists in performing a best fit estimation of J as a function of E by a function g belonging to a chosen family. For example, we may choose piecewise linear functions or polynomial functions of small degree. We have performed two such estimations g_1 and g_2 of J by the means of the experimental values of $\overline{J}$ as a function of $\overline{E}$.

▷ A rough best fit estimation by a polynomial function of degree 2 yields $g_1(x) = x^2$;

▷ A rough best fit estimation by a simple piecewise linear function yields

$$g_2(x) = \begin{cases} 1/10x & \text{if } x \in [0, 0.01], \\ 1/3x + 7/300 & \text{if } x > 0.01 \end{cases}$$

Denote by f_1 and f_2 primitives of g_1 and g_2 respectively. Experiments show that the two distorded cost functions $f_1(E)$ and $f_2(E)$ yield similar very good results up to several hundred vertices. Therefore, we see that a more accurate best fit estimation of J would not be useful. Experimental results are shown in Table II. The obtained speedup was 4.5 with a network of 13 Transputers.

TABLE II
Statistics over 20 executions with randomly generated Hamiltonian graphs of order n, function "Distorted Mean".

$N_{acc} = \frac{n^2}{2}$	$n = 100$	$n = 200$	$n = 500$
sucess ratio	1	1	1
average final value of $l(F)$	99	199	499
$N_{acc} = \frac{n^2}{10}$			
sucess ratio	1	1	1
average final value of $l(F)$	99	199	499

6. CONCLUSION

We have presented some problem-independent amelioration techniques for the Simulated Annealing algorithm. The study of cycles provides a convenient theoretical framework leading to sharp evaluations for the speed of convergence of the algorithm, as well as to the important practical method of distortions of the cost function.

Numerous exact parallelizations of the Simulated Annealing algorithm have been proposed. We have highlighted three main families, the Multiple Trials strategies, the Multi-Temperature strategies and the Speculative strategies.

Speculative algorithms are highly scalable. However several difficulties arise with speculative strategies, concerning the management of the tree of processors and the load balancing. They deserve further experiments.

Multi-Temperature strategies seem to be well-suited to a relatively small number of processors. Nevertheless, they yield interesting results when applied to hard problems. Further experiments remain to be done, concerning the fine tuning of the temperatures.

Multiple Trials Annealings have been experimented by many researchers, and successfully applied to various optimization problems. They allow lots of variants. One can find such variants well-suited to numerous particular problems. When the move evaluation time is very large compared to the move communication time, one can use an Unbiased Synchronous by Move strategy or a Synchronous by Trial strategy at high temperature, and then a Synchronous by Move one as the temperature lowers. One can, even, use a processor farm if the move generation time is small enough. When the ratio between the move evaluation time and the communication time is not so favorable, on can use strategies synchronizing after $a > 1$ moves, thereby allowing bounded desynchronizations. Another approach is the variable granularity strategy progressively sliding from a Synchronous by Trial strategy to an Unbiased Synchronous by Move one as the temperature decreases. There are limits in the extensibility of the major part of the Multiple Trials strategies, the variable granularity one is a good candidate at pushing these limits forward.

It is possible to design parallel libraries facilitating the implementation of Parallel Annealings, as well as Parallel Taboo Search algorithms, on various parallel architectures [11]. They provide flexible and general tools, allowing operational research practitioners to safely use parallel architectures. These libraries are easily extensible to other parallelizations of simulated annealing like asynchronous ones or combinations with data parallelism. Moreover, this approach suggests an interesting direction of research concerning the comparison between Taboo Search and Simulated Annealing used in combination with cost function distortion, when applied to large size optimization problems. Another interesting direction of research concerns mixed strategies involving approximate methods such as Simulated Annealing and Tabu Search as well as exact ones such as Branch and Bound.

ACKNOWLEDGEMENTS

This work has greatly benefited from discussions with Robert Azencott and Olivier Catoni. We thank the anonymous referee for useful suggestions.

REFERENCES

1. E. H. L. Aarts, M. J. de Bont, E. H. A. Habers and P. J. M. van Laarhoven, Parallel implementation of the statistical cooling algorithm, *The VLSI Journal*, June 1986, *4*, pp. 209-238.
2. E. H. L. Aarts and P. J. M. van Laarhoven, *Simulated Annealing and Applications: Mathematics and its Applications*, D. Reidel Publishing Company, 1987.
3. J. Akayama, G. Exoo and F. Harary, Covering and packing in graph III: cyclic and acyclic invariant, *Math. Slovaca*, 1980, *30*, pp. 405-417.
4. E. H. L. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machines*, John Wiley & Sons Ltd., 1989.
5. R. Azencott, *Sequential Annealing: Acceleration by Monotone Concave Distortions of the Energy Function*, section 1.5, pp. 6-7, In Azencott [6], 1992.
6. R. Azencott, editor, *Simulated Annealing: Parallelization Techniques*, John Wiley & Sons inc., 1992.
7. O. Catoni, *Etude Asymptotique des Algorithmes de Recuit Simulé*, PhD thesis, Université d'Orsay, 1990.
8. O. Catoni, *Rates of Convergence for Sequential Annealing: a Large Deviation Approach*, chapter 3, pp. 25-36, In Azencott [6], 1992.
9. T. S. Chiang and Y. Chow, On the convergence rates of annealing processes, *SIAM J. Control Optimization*, 1988, *26*, pp. 1455-1470.
10. D. Delamarre, J. L. Fouquet, H. Thuillier and B. Virot, Linear-k-arboricity of cubic graphs. Research Report 93-8, Laboratoire d'Informatique Fondamentale d'Orléans, Université d'Orléans BP 6759 F45067 Orléans Cedex 2, 1993.
11. D. Delamarre, *Etude et Conception d'Algorithmes Parallèles d'Optimisation Combinatoire Discrète Approchée. Implantation sur Architectures MIMD*, PhD thesis, Université d'Orléans, LIFO, 4, rue Léonard de Vinci, BP 6759 F45067 Orléans Cedex 2, France, Déc. 1994.
12. D. Delamarre and B. Virot, *Simulated Annealing Applied to the Hamiltonian Problem for Cubic Graphs*, Research Report 93-3, Laboratoire d'Informatique Fondamentale d'Orléans, Université d'Orléans, BP 6759 F45067 Orléans Cedex 2, France, 1993.
13. R. Diekmann, R. Lülling and J. Simon, A general purpose distributed implementation of simulated annealing. In *IMACS'91 13th World Congress on Computation and Applied Mathematics*, 1991, pp. 816-817.
14. M. D. Durand, *Trading Accuracy for Speed in Parallel Simulated Annealing Algorithms*, PhD thesis, Columbia University, 1990.
15. M. D. Durand and S. R. White, Permissible error in parallel simulated annealing, Technical report, Institut de Recherche en Informatique et Systèmes aléatoires, Rennes, France, July 1991.
16. R. W. Eglese, Simulated annealing: a tool for operational research, *European Journal of Operational Research*, 1990, *46*, pp. 271-281.
17. M. I. Friedlin and A. D. Wentzel, *Random Perturbations of Dynamical Systems*, Springer-Verlag, 1984.
18. M. R. Garey and M. R. Johnson, *Computer and Intractability*, Freeman, 1979.
19. G. Graffigne, *Parallel Annealing by Periodically Interacting Multiple Searches: an Experimental Study*, chapter 5, pp. 47-79, In Azencott [6], 1992.
20. D. R. Greening, Parallel simulated annealing techniques, *Physica*, 1990, D, *42*, pp. 293-306.
21. D. R. Greening, Simulated annealing with inacurrate costs functions, unpublished, October 1993.
22. B. Hajek, Cooling schedules for optimal annealing, *MOR*, 1988, *13*, pp. 311-329.

23. B. HAJEK and G. H. SASAKI, The time complexity of maximum matching by simulated annealing, *J. of the ACM*, 1988, *35*, pp. 387-403.
24. L. INGBER, Very fast simulated reannealing, *Journal of Mathematical Computing Modelling*, 1989, *12 (8)*, pp. 967-973.
25. L. INGBER, Simulated annealing: Practice *versus* theory, *Journal of Mathematical Computing Modelling*, May 1993, *18 (11)*, pp. 23-57
26. K. KIMURA and K. TAKI, Time-homogeneous parallel annealing algorithm. In *IMACS' 91 13th World Congress on Computation and Applied Mathematics*, 1991, pp. 827-828.
27. S. KIRKPATRICK, C. D. GELATT Jr and M. P. VECCHI, Optimization by simulated annealing, *Science*, May 1983, *220 (4598)*, pp. 671-680.
28. R. H. J. M. OTTEN, L. P. P. P. van GINNEKEN, *The Annealing Algorithm*, Kluwer Academic Publisher, 1989.
29. C. R. REEVES, *Modern Heuristic Techniques for Combinatorial Problems*, Colin R. Reeves, blackwell scientific publications editions, 1993, Osney Mead, Oxford OX2 0EL.
30. P. ROUSSEL-RAGOT and G. DREYFUS, *Parallel Anealing by Multiple Trials: an Experimental study on a Transputer Network*, chapter 7, pp. 91-108, In AZENCOTT [6], 1992.
31. B. VIROT, *Parallel Annealing by Multiple Trials: Experimental study of a Chip Placement Problem Using a Sequent Machine*, chapter 8, pp. 109-129, In Azencott [6], 1992.
32. E. E. WHITE, R. D. CHAMBERLAIN and M. A. FRANKLIN, Parallel simulated annealing using speculative computation, *IEEE Transactions on Parallel and Distributed Systems*, October 1991, *2 (4)*.
33. S. R. WHITE, Concepts of scale in simulated annealing, In *IEEE International Conference on Computer Design*, 1984, pp. 646-651.