SIAM J. CONTROL OPTIM.
Vol. 36, No. 5, pp. 1539–1575, September 1998

003

SOLVING SCHEDULING PROBLEMS BY SIMULATED ANNEALING*

OLIVIER CATONI†

Abstract. We define a general methodology to deal with a large family of scheduling problems. We consider the case where some of the constraints are expressed through the minimization of a loss function. We study in detail a benchmark example consisting of some jigsaw puzzle problem with additional constraints. We discuss some algorithmic issues typical of scheduling problems, such as the apparition of small unused gaps or the representation of proportionality constraints. We also carry on an experimental comparison between the Metropolis algorithm, simulated annealing, and the iterated energy transformation method to see whether asymptotical theoretical results are a good guide towards practically efficient algorithms.

Key words. scheduling problems, Metropolis algorithm, simulated annealing, IET algorithm

AMS subject classifications. 60J10, 90C42, 82C80, 65C05

PII. S0363012996307813

Introduction. The aim of this paper is to describe a general strategy to deal with scheduling problems and to illustrate its use on the resolution of jigsaw puzzles. We will assume that we can put our scheduling problem in the form of a task assignment problem, and we will turn it into the minimization of a cost function defined on a suitable search space. This cost function will be minimized by a Monte Carlo algorithm of the Metropolis kind: either simulated annealing or our recently introduced iterated energy transformation (IET) method. We have already studied some of the theoretical aspects of these two methods in previous papers (see [6], [7]).

We have chosen to experiment on a jigsaw puzzle problem with rectangular pieces because this is a typical instance of the kind of difficulties encountered when building time tables, and because it is in itself a difficult problem (it is NP-complete) which deserves special attention. In the course of this experimentation, we will compare four algorithms: a randomized descent algorithm (the Metropolis dynamic at temperature zero), the Metropolis algorithm, simulated annealing, and the iterated energy transformation algorithm.

1. An abstract task assignment framework. Let B be a finite set of tasks. Let E be a set of resources needed to perform these tasks. The set E may be any kind of set, a finite set, a domain in $\mathbb{R}^n$, etc. In applications it can represent various things, such as a set of people who are to perform the tasks, in which case it is natural to see it as a finite set, or it can also represent space and time needed for the tasks, in which case it is sometimes natural to see it as a domain in $\mathbb{R}^n$. More often it is a product space of both kinds. Anyhow, we will only consider a finite collection of subsets of E; therefore it will always be possible to consider that E is a finite set from the theoretical point of view. This is reasonable, because a computer can only handle a finite number of possible ways to allocate resources and also because, in many problems of the time-table type, continuous quantities, such as time, are discretized (for instance when one tries to schedule lectures, they are usually constrained to start at full hours). Anyhow, the reader should think of E as a large set and our methods,

*Received by the editors August 5, 1996; accepted for publication (in revised form) August 20, 1997; published electronically June 3, 1998.
http://www.siam.org/journals/sicon/36-5/30781.html
†D.I.A.M. - Laboratoire de Mathématiques de l'Ecole Normale Supérieure, U.A. 762 du C.N.R.S., 45 rue d'Ulm, 75005 Paris, France (catoni@dmi.ens.fr).

inherited from statistical mechanics, are precisely meant to cope with a large state space.

The abstract scheduling problem we will consider is to allocate to each task in B a set of resources in a way which satisfies a set of constraints.

At this level of generality, we will not represent the constraints by equations or logical relations; we will merely view them as a subset $\mathcal{S}$ of $\mathcal{P}(B \times E)$ (where $\mathcal{P}(A)$ is the set of subsets of the set A). We will call $\mathcal{S}$ the "solution space." A solution x in $\mathcal{S}$ is a subset of the product space $B \times E$. We will use the notations π_B and π_E for projections on B and E. The fact that $(b, e) \in x$ means that the task b uses the resource e. The set of resources used by b is $\pi_E(\pi_B^{-1}(b) \cap x)$, for which we will use the functional notation $x(b)$.

We will assume that each solution $x \in \mathcal{S}$ is a complete assignment, in the sense that all the tasks are scheduled:

$$\pi_B(x) = B \quad \text{for any } x \in \mathcal{S}.$$

Our scheduling problem is to construct a solution x belonging to the solution space $\mathcal{S}$.

The idea of considering scheduling problems as putting objects in boxes in a multi-dimensional space is not new and can be found, for instance, in Abramson [1], where a specialized simulated annealing hardware is described for handling some generic types of cost functions.

2. The jigsaw puzzle example. This example is meant to be a benchmark, where the main algorithmic issues of scheduling problems are present.

The set of resources E will be a discretized rectangular frame

$$E = \{0, \dots, M-1\} \times \{0, \dots, N-1\} \subset \mathbb{Z}^2.$$

The set of tasks B will be the set of pieces of the jigsaw puzzle. Each piece r has a rectangular shape defined by its width $w_r \in \mathbb{N}^*$ and by its height $h_r \in \mathbb{N}^*$. The constraint is that pieces should not overlap. Thus the solution space is

$$\mathcal{S} = \{x \subset B \times E \ : \ x(r) = [a_r, a_r + w_r[\times[b_r, b_r + h_r[, \quad (a_r, b_r) \in E, \quad r \in B, \\ \text{and } x(r) \cap x(r') = \emptyset, \quad r \neq r' \in B\}.$$

The problem is to build the jigsaw puzzle; that is, to construct $x \in \mathcal{S}$. Although the shape of pieces is very simple, this problem can be seen to be very complex. In fact, it is easy to see that it is NP complete, because it contains the partition problem among its instances (see [14]). Indeed, the partition of given integers $\{c_1, \dots, c_N\}$ into two sets I and J such that

$$\sum_{i \in I} c_i = \sum_{j \in J} c_j$$

can be viewed as a jigsaw puzzle with N pieces, respectively, of width c_i and height 1, and a frame of width $(1/2) \sum_{i=1}^N c_i$ and height 2 (see Fig. 2.1).

3. A method of resolution based on the Metropolis dynamic. In this section we will sketch a methodology to solve the abstract problem of section 1. The general idea is to perform a random search for a solution in a state space larger than the solution space. This search space should be easy to describe and easy to search

FIG. 2.1.

by a Markov chain performing a succession of elementary moves. Of course, we will not use a Markov chain which uniformly samples the search space because, usually, the search space we will be able to build will be very large when compared to the solution space, and drawing points at random in the search space would seldom lead to discovering a solution.

Instead we will use a Markov chain with rare transitions, whose invariant measure is concentrated in a neighborhood of the solution space. This optimization technique is well known, but its improvement is still a subject of active research. The prototype algorithm we will start from is the Metropolis dynamic at low temperature. The Metropolis dynamic has been designed to simulate statistical mechanics systems, and not for optimization purposes. In order to improve its performance as an optimization algorithm, some speed-up techniques have been proposed. The most famous one is simulated annealing [15], [18]. We have also proposed recently another technique, which we called *the iterated energy transformation method* (IET) [7]. We will describe and use both of these.

3.1. Choice of a search space. The first step of the method is to choose a search space $\tilde{\mathcal{S}}$ containing the solution space $\mathcal{S}$. The most popular way to construct $\tilde{\mathcal{S}}$ is to relax some constraints about the solution and to measure, instead, how much the constraints have been violated by a score function one has afterwards to minimize. For instance, in circuits placement applications (one of the earliest applications of simulated annealing) the constraint that circuits should not overlap is often relaxed, and the overlapping of circuits is instead merely discouraged by some score function of the surface of the overlap. Our strategy will be somewhat of the same kind, with the difference that we will not relax a constraint which is specific to the problem. Instead, we will allow partial solutions, where only some proportion of the tasks have been scheduled. Defining partial solutions is usually very easy and very natural. Most of the time, this is how the problem is posed from the beginning. Indeed the constraints come usually from incompatibilities between tasks, such as sharing the same resource or needing to be performed in a given order, and can be expressed without assuming that all the tasks are already scheduled.

From the technical point of view, we will assume that the search space (the space of partial solutions) satisfies the following properties:

- The empty solution is in the search space: $\emptyset \in \tilde{\mathcal{S}}$.
- There is a path from the empty solution leading to any partial solution $x \in \tilde{\mathcal{S}}$ along which tasks are scheduled one after the other. This can be expressed in the following way:

  For any $x \in \tilde{\mathcal{S}}, x \neq \emptyset$, there is $b \in \pi_B(x)$ such that $x \setminus \pi_B^{-1}(b) \in \tilde{\mathcal{S}}$.
- All complete solutions in the search space satisfy the constraints. In other words, the solution space is exactly made of the complete solutions of the

search space. This is expressed by the following equation:

$$\mathcal{S} = \{x \in \tilde{\mathcal{S}} \ : \ \pi_B(x) = B\}.$$

Let us notice that the "best" choice for $\tilde{S}$ would be $\{x \cap \pi_B^{-1}(C) \ : \ x \in \mathcal{S}, C \subset B\}$, the set of all partial solutions contained in global solutions. Anyhow, this set is, in practical situations, never defined by simple relations, because when you have scheduled some of the tasks, it is never possible (except for trivial problems) to foretell whether there will remain suitable resources to schedule the remaining ones. Therefore the search space $\tilde{S}$ is, most of the time, much broader than $\mathcal{S}$ and contains many dead ends.

3.2. Building the dynamic: Constructions and destructions. The next idea is to define on $\tilde{S}$ two kinds of dynamics, a *constructive dynamic* and a *destructive dynamic*. These two random dynamics are characterized by two Markov matrices q_C and q_D,

$$\begin{aligned} q_C &: \tilde{\mathcal{S}} \times \tilde{\mathcal{S}} \longrightarrow [0,1], \\ q_D &: \tilde{\mathcal{S}} \times \tilde{\mathcal{S}} \longrightarrow [0,1]. \end{aligned}$$

We will assume that the transitions allowed by q_C consist of either keeping the current partial solution or scheduling one more task. In a similar way the transitions allowed by q_D consist of unscheduling a given number of tasks. We allow unscheduling of more than one task at a time, because it is in some situations more sensible to do so. For instance, if many tasks have to share the same resource, it may sometimes speed up the allocation process to unschedule all of them at the same time (think of students sharing the same teacher).

This conception of constructions and destructions can be expressed by the following equations, where $|A|$ is the number of elements in the finite set A:

$$\left\{ \begin{array}{l} \bullet \ \{(x,y) \ : \ q_C(x,y) > 0, x \neq y\} \\ \qquad = \{(x,y) \ : \ y \cap \pi_B^{-1}(\pi_B(x)) = x, |\pi_B(y)| = |\pi_B(x)| + 1\}, \\ \bullet \ \{(x,y) \ : \ q_C(y,x) > 0, x \neq y\} \\ \qquad\qquad \subset \{(x,y) \ : \ q_D(x,y) > 0\} \\ \qquad\qquad\qquad \subset \bigcup_{n=1}^{+\infty} \{(x,y) \ : \ q_C^n(y,x) > 0\}. \end{array} \right. \tag{1}$$

Let us remark that, usually, constructions will decompose into two steps, one being to choose an unscheduled task $b \in B \setminus \pi_B(x)$ and the second one being to try to allocate a set of resources to it. This second step is sometimes unsuccessful (either because it is impossible or the proper allocation has not been discovered); therefore as a rule, we have $q_C(x,x) > 0$ for a substantial number of partial solutions. On the contrary, destructions are simple moves, where you have only to choose a scheduled task b in $\pi_B(x)$ and to remove it. Therefore as a rule, we will have $q_D(x,x) = 0$, except when $x = \emptyset$, for which $q_D(\emptyset, \emptyset) = 1$.

When the two above assumptions are satisfied, the whole search space $\tilde{\mathcal{S}}$ can be constructed by q_C starting from the empty solution $\emptyset$, and reversely, any solution can

be shrunk to the empty solution by successive applications of q_D. More precisely, the following proposition holds.

PROPOSITION 3.1.

$$\begin{aligned}\tilde{\mathcal{S}} &= \bigcup_{n=0}^{+\infty} \{y \ : \ q_C^n(\emptyset, y) > 0\} \\ &= \bigcup_{n=0}^{+\infty} \{x \ : \ q_D^n(x, \emptyset) > 0\}.\end{aligned}$$

3.3. Building the cost function. Now we will build a cost, or energy, function defined on the search space, which penalizes partial solutions: we will call it $U \ : \ \tilde{\mathcal{S}} \longrightarrow \mathbb{R}$. Namely, we will require the following properties to hold:

(2)
$$\begin{cases} \bullet \ \arg\min_{x \in \tilde{\mathcal{S}}} U(x) = \mathcal{S}. \\ \bullet \ \text{There is a positive constant } \gamma \text{ such that } U(y) \geq U(x) + \gamma \text{ when } x \neq y \text{ and } q_D(x, y) > 0. \end{cases}$$

A typical example for U is

$$U(x) = \mu(B \setminus \pi_B(x)),$$

where μ is a positive measure on B. In this case the assumptions on U are satisfied and the largest choice of γ is

$$\gamma = \min_{b \in B} \mu(b).$$

3.4. Building a Metropolis dynamic. From Proposition 3.1, we see that any Markov matrix of the form $\lambda q_C + (1 - \lambda) q_D$ with $\lambda \in]0, 1[$ is irreducible. Therefore a straightforward way to build a Metropolis dynamic would be to consider the Markov matrix

$$p_T(x, y) = \begin{cases} (\lambda \, q_C(x, y) + (1 - \lambda) \, q_D(x, y)) e^{-(U(y) - U(x))^+ / T}, & x \neq y \\ 1 - \sum_{z \neq x} p_T(x, z), & x = y. \end{cases}$$

In fact, we can do better because we know in advance that, during a construction, the energy will decrease, and that during a destruction, the energy will increase by a quantity at least equal to γ. This avoids applying uselessly the kernel q_D at low temperatures in situations where we know that it will, most of the time, generate a move to be rejected.

More precisely, we will use the following Markov matrix:

(3)
$$\begin{aligned} p_T(x, y) &= \lambda \ e^{-\gamma / T} q_D(x, y) e^{-(U(y) - U(x) - \gamma)^+ / T} \\ &\quad + q_C(x, y) \left(1 - \lambda \ \sum_{z \in \tilde{\mathcal{S}}} q_D(x, z) e^{-(U(z) - U(x) - \gamma)^+ / T - \gamma / T} \right), \end{aligned}$$

where λ is again a positive parameter in the interval $0 < \lambda \leq 1$. A choice of $\lambda < 1$ avoids that destructions should always be chosen at high temperatures. Usually we will take $\lambda = 1/2$ or $\lambda = 1$. The positive part in $(U(y) - U(x) - \gamma)^+$ is needed only to cover the case where $x = y$.

The computer implementation of this Metropolis dynamic is the following: starting from the state x,

- first flip a coin with odds $\lambda e^{-\gamma/T}$ and $1 - \lambda e^{-\gamma/T}$ to decide whether or not to try a destruction.
- in case a destruction is tried,
  - choose a transition (x, y), drawing y according to the probability distribution $q_D(x, y)$.
  - then flip a second coin with odds $\exp -((U(y) - U(x) - \gamma)^+/T)$ and $1 - \exp -((U(y) - U(x) - \gamma)^+/T)$ to decide whether or not to apply this move.
- if the answer to one of the two previous tosses was *no*, then choose a transition (x, y), where y is chosen according to the distribution $q_C(x, y)$, and apply it.

The hypotheses we made about q_C, q_D, and U are what are needed to prove the following proposition.

PROPOSITION 3.2. *For any temperature $T > 0$, the matrix p_T is an irreducible Markov matrix.*

Considering the rate function $V : \tilde{\mathbb{S}} \times \tilde{\mathbb{S}} \to \mathbb{R}_+ \cup \{+\infty\}$ defined by

$$V(x, y) = \begin{cases} (U(y) - U(x))^+ & \text{if } q_D(x, y) + q_C(x, y) > 0 \text{ and } x \neq y, \\ +\infty & \text{otherwise,} \end{cases}$$

we see that there is a positive constant κ such that, whenever $x, y \in \tilde{\mathbb{S}}$, $x \neq y$,

$$\kappa\, e^{-V(x,y)/T} \leq p_T(x, y) \leq \frac{1}{\kappa}\, e^{-V(x,y)/T}.$$

Moreover V satisfies the weak reversibility condition of Hajek–Trouvé with respect to U. More precisely, if $\Gamma_{x,y}$ is the set of paths from x to y, we put for any $\gamma = (\gamma_1 = x, \ldots, \gamma_r = y) \in \Gamma_{x,y}$

$$H(\gamma) = \max_{i=1,\ldots,r-1} U(\gamma_i) + V(\gamma_i, \gamma_{i+1})$$

and

$$H(x, y) = \min_{\gamma \in \Gamma_{x,y}} H(\gamma).$$

The weak reversibility condition of Hajek–Trouvé states that for any $x, y \in \tilde{\mathbb{S}}$

$$H(x, y) = H(y, x).$$

Due to this reversibility property, U is a quasi-potential for p_T. We mean by this statement that the (unique) invariant probability measure μ_T of p_T satisfies for some positive constant α (independent of T) and for any $x \in \tilde{\mathbb{S}}$

$$\alpha \leq \mu_T(x) e^{(U(x) - \min U)/T} \leq 1/\alpha.$$

COROLLARY 3.1. *We can build optimization algorithms based on p_T following the results of Catoni* [7] *and Trouvé* [22]. *More precisely, for any fixed value of the temperature T, the homogeneous Markov chain with transition matrix p_T is a generalized Metropolis algorithm with quasi-potential function U. In the same way, for any decreasing sequence of temperatures $(T_n)_{n\in\mathbb{N}}$, the nonhomogeneous Markov chain $(X_n)_{n\in\mathbb{N}}$ on $\tilde{\mathcal{S}}$ with transitions*

$$P(X_n = y \ : \ X_{n-1} = x) = p_{T_n}(x, y)$$

is a generalized simulated annealing algorithm. Its behavior has been studied in [22] *and* [24] *and is very similar to the behavior of classical simulated annealing as studied in* [6].

We can also apply the iterated energy transformation method to p_T, which will be described in a further section of this paper and is studied in [7].

Proof. The only nonstraightforward point to check is the Hajek–Trouvé weak reversibility condition. Let us consider $x, y \in \tilde{\mathcal{S}}$, and $\gamma \in \Gamma_{x,y}$. We build a path from y to x in the following way. Replace any edge $(z,t) \in \gamma$ by the edge (t,z) if $q_C(z,t) > 0$ or $p_T(z,t) = 0$. If neither of the above two conditions is true, this means that $q_D(z,t) > 0$; then there is a path $\varphi \in \Gamma_{t,z}$ such that $q_C(u,v) > 0$ for any edge $(u,v) \in \varphi$, and we replace (z,t) by φ. The path φ is such that $H(\varphi) = U(z) + V(z,t) = U(t)$ because for any $(u,v) \in \varphi$, $U(v) < U(u)$ and $V(u,v) = 0$. Therefore by concatenating all these reversed edges and paths in reverse order we get a path $\psi \in \Gamma_{y,x}$ such that $H(\psi) = H(\varphi)$. Therefore $H(y,x) = H(x,y)$. □

Remarks.

- In the search space we consider, there is a natural starting point for optimization algorithms, which is the empty schedule $\emptyset$.
- In many scheduling problems, it is not known in advance whether a complete solution exists or whether one can possibly be found within the available computer time. Our method has the advantage of finding at least a partial solution, where some proportion of the tasks are scheduled in a coherent way. This is not the case if other constraints are relaxed, as is usually done. For instance, if the aim is to schedule the lectures at a university, a solution where some lectures share the same room at the same time has no practical interest, whereas a solution where some proportion of the lectures are scheduled in a coherent way can be applied.
- A slight variant of the present setup is the case where the search space satisfies condition (1), but one does not know whether it is possible to schedule all the tasks, and wants instead to schedule as many tasks as possible. In this situation the energy can weigh (through a positive measure) the relative importance of tasks.

In the three following sections, we are going to recall briefly some theoretical results about the speed of convergence of three optimization algorithms.

3.5. Rate of convergence of the Metropolis algorithm. In this section we consider the canonical process $(X_n)_{n\in\mathbb{N}}$ on the canonical space $(\tilde{\mathcal{S}}^{\mathbb{N}}, \mathcal{B})$, where $\mathcal{B}$ is the sigma field generated by the events depending on a finite number of coordinates.

For any temperature $T \in \mathbb{R}_+$, P_T will be the probability distribution on $(\tilde{\mathcal{S}}^{\mathbb{N}}, \mathcal{B})$ of a Markov chain with transition matrix p_T (where p_T is as in Proposition 3.2). Under this distribution, $(X_n)_{n\in\mathbb{N}}$ is a Metropolis algorithm and has the following convergence speed.

PROPOSITION 3.3. *There exists a positive constant d, depending only on the choice of the search space $\tilde{\mathcal{S}}$, of the constructive and destructive dynamics q_C and q_D and of the parameter λ, $0 < \lambda < 1$, such that for any energy function U satisfying the hypothesis* (2) *of section* 3.3, *for any positive constant η,*

$$\max_{x\in\tilde{\mathcal{S}}} P_T(U(X_N) \geq U_{\min} + \eta \,|\, X_0 = x) \leq d\left(\exp - \left(\frac{N}{d} e^{-H_1/T}\right) + e^{-\eta/T}\right),$$

where $H_1(V)$ is the first critical depth of the rate function V defined in Proposition 3.2. *The exponent $H_1(V)/T$ is optimal when η is small and when T tends to* 0 *and N tends to $+\infty$. With the notations of Proposition* 3.2,

$$H_1(V) = \max_{x\notin\mathcal{S}} \min_{y\in\mathcal{S}} H(x,y) - U(x).$$

As a consequence, considering $1/T = (1/H_1)\log(N\, H_1/d\,\eta\, \log N)$, we see that there is a constant d (independent of U and η), such that

$$\inf_{T\in\mathbb{R}_+} \max_{x\in\tilde{\mathcal{S}}} P_T(U(X_N) \geq U_{\min} + \eta \,|\, X_0 = x) \leq d\left(\frac{d\,\eta}{H_1(V)}\,\frac{\log N}{N}\right)^{\eta/H_1(V)}.$$

Moreover the exponent $\eta/H_1(V)$ is optimal for small enough values of $\eta \in (U(E) - U_{\min})$.

For a proof see, for instance, Cot and Catoni [9].

This proposition does not give a quantitative upper bound for the probability of failure for N iterations, since it does not give an estimate for constant d; nevertheless, the fact that this constant is independent of U allows us to compare the probability of failure for different energy functions U for a large *finite* number of iterations N and not only when N tends to infinity. More precisely, it shows that the convergence speed of the Metropolis algorithm is slow when there are states with energies close to $U_{\min}$. Indeed if one wants to study the convergence to $\mathcal{S}$, one has to choose

$$\eta = \min\{U(x) - U_{\min}\,,\, x \in \tilde{\mathcal{S}} \setminus \mathcal{S}\}.$$

If η is small, then it will reflect on the exponent $\eta/H_1(V)$.

This is a theoretical justification for the introduction of simulated annealing, which will not suffer from this drawback, when proper robust cooling schedules are used.

3.6. Rate of convergence of simulated annealing. We consider now a non-increasing triangular sequence $T_1^N \geq T_2^N \geq \cdots \geq T_N^N$ of temperatures and the measure $P_{(T_1^N,\ldots,T_N^N)}$ on $\tilde{\mathcal{S}}^N$ of the nonhomogeneous Markov chain with transitions

$$P_{(T_1^N,\cdots,T_N^N)}(X_n = y \,|\, X_{n-1} = x) = p_{T_n^N}(x,y).$$

The rate of convergence of such an algorithm has been studied in [6], [8], and [22] (see also [24] in English, translated from [22]). We give here a simple result; for more precise estimates, we refer to the original papers.

PROPOSITION 3.4 ([6], [22], [24]). *There is a positive constant K such that*

$$K^{-1}N^{-D^{-1}} \leq \inf_{T_1^N \geq \cdots \geq T_N^N} \max_{x \in \tilde{\mathcal{S}}} P_{(T_1^N,\ldots,T_N^N)}(U(X_N) > U_{\min} \mid X_0 = x) \leq KN^{-D^{-1}},$$

where the constant $D = D(V)$ is the difficulty of the rate function V. With the notations of Proposition 3.2, *the definition of $D(V)$ is*

$$D(V) = \max_{x \in \tilde{\mathcal{S}} \setminus \mathcal{S}} \min_{y \in \mathcal{S}} \frac{H(x,y) - U(x)}{U(x) - \min U}.$$

For any $A > 0$, there is a positive constant K such that the triangular exponential schedule

$$T_n^N = \frac{1}{A}\left(\frac{A}{(\log N)^2}\right)^{n/N}$$

gives a convergence speed of

$$\max_{x \in \tilde{\mathcal{S}}} P_{T^N}(U(X_N) > U_{\min} \mid X_0 = x) \leq K\left(\frac{\log N \, \log\log N}{N}\right)^{D(V)^{-1}},$$

for N large enough.

In the case of simulated annealing, we have a probability of failure for N iterations of order $\epsilon \asymp_{\log} (1/N)^{1/D}$ (meaning that the logarithms on both sides of this equation are equivalent when N tends to infinity). The important feature of this theoretical result is that the exponent $1/D$ is independent of the precision with which we want to reach $U_{\min}$ but depends, on the contrary, only on the structure of the local minima of U.

One other interesting point is that the exponential triangular cooling schedule $T_n^N = A^{-1}(A/(\log N)^2)^{n/N}$ is robust: it gives a convergence rate with the optimal exponent $1/D(V)$ for any energy function U.

3.7. Rate of convergence of the energy transformation method. We introduced in [7] the iterated energy transformation method as another mean to discourage uphill moves from low energy states more than from high energy states. In simulated annealing this effect is produced by an exogenous control of the temperature parameter: in "typical" successful runs of simulated annealing, the energy of the current state is moving downwards on the average, and at the same time uphill moves are more and more discouraged. In the iterated energy transformation method, a temporary hypothesis is made about the value of $U_{\min}$, and a concave transformation is applied to U on the basis of this hypothesis. Then the algorithm is run at constant temperature using the transformed energy. This produces the desired effect of discouraging more uphill moves from low energy states. Of course, in the beginning, the hypothesis about $U_{\min}$ is necessarily grossly underestimated, so that the energy transform is not very efficient, but after some iterations, it can be improved (this will work with a probability close to one) depending on the values of the energies of the explored states.

The convergence of the lower bound estimate for $U_{\min}$ towards the true value of $U_{\min}$ is exponentially fast (with a probability close to one), and therefore the energy transformation is quickly tuned to an efficient value.

The iterated energy transformation method applied to our problem is described as follows. For any strictly concave, strictly increasing energy transformation $F : \mathbb{R} \longrightarrow \mathbb{R} \cup \{-\infty\}$, we consider the Markov matrix

$$p_F(x,y) = \mathbf{1}_{(F(x)>-\infty)} \Bigg\{ \lambda e^{(F(U(x))-F(U(x)+\gamma))} q_D(x,y) e^{(F(U(x)+\gamma)-F(U(y)))^-} + q_C(x,y) \left(1 - \lambda \sum_z q_D(x,z) e^{F(U(x))-F(U(x)+\gamma)+(F(U(x)+\gamma)-F(U(z)))^-} \right) \Bigg\} + \mathbf{1}_{(F(x)=-\infty)} \mathbf{1}_{(x=y)}.$$

Consider for any positive constant α, any real shift τ, and any positive temperature T, the transformation

$$F_{\alpha,T,\tau}(U) = \begin{cases} \alpha\, U + \dfrac{1}{T} \log(U+\tau) & \text{if } U+\tau > 0, \\ -\infty & \text{otherwise .} \end{cases}$$

Let us introduce the simplified notation $\overline{p}_{\alpha,T,\tau} = p_{F_{\alpha,T,\tau}}$.

Given parameters $M \in \mathbb{N}$ (number of iterations performed with each energy transform), two real numbers $\rho > 0$ and $\eta_0 \geq 0$ (two parameters for the update of the shift τ), and an initial lower bound $\delta < U_{\min}$, we consider the canonical process $(X_n)_{n\in\mathbb{N}}$ on $\tilde{\mathcal{S}}^{\mathbb{N}}$ with probability distribution $P_{\alpha,T,M,\rho,\eta_0}$ defined by the following conditional distributions:

$$\begin{aligned} P_{\alpha,T,M,\rho,\eta_0}(X_n = y \,|\, (X_0,\dots,X_{n-1}) = (x_0,\dots,x_{n-1})) \\ = \overline{p}_{\alpha,T,\tau_n(x_0,\dots,x_{n-1})}(x_{n-1}, y), \end{aligned}$$

with

$$\begin{cases} \tau_r &= \eta_0 - \delta, \quad 0 < r \leq M, \\ \tau_{kM+r} &= \tau_{k\,M} - \dfrac{1}{1+\rho} \left(\min\limits_{n, n\leq k\,M} U(X_n) + \tau_{k\,M} \right) + \eta_0, \quad 0 < k, 0 < r \leq M. \end{cases}$$

We have proved in [7] the following theorem.

THEOREM 3.1 (Catoni). *For any fixed $\alpha > 0$, the family of processes described above satisfies for some positive constants B and K, for any choice of $r \in \mathbb{N}$, $\eta_0 \geq 0$, $\rho > 0$,*

$$\max_{x\in\tilde{\mathcal{S}}} P_\theta(U(X_{r\,M}) \geq \eta \,|\, X_0 = x) \leq \epsilon$$

where $\theta = (\alpha, T, M, \rho, \eta_0)$,

$$\begin{aligned} T &= \frac{\log(1+\rho)}{\log(Kr/\epsilon)} \\ M &= B \left(\frac{\epsilon}{K\,r} \right)^{-\log(1+\tilde{D}_{\eta_0})/\log(1+\rho)} \log \frac{K\,r}{\epsilon}, \\ &= \frac{B}{T} \log(1+\rho) \left(1 + \tilde{D}_{\eta_0} \right)^{1/T}, \\ \eta &= U_{\min} + \rho \left(\frac{\rho}{1+\rho} \right)^{r-1} (U_{\min} - \delta + \eta_0) + \eta_0 \rho (1+\rho), \end{aligned}$$

and where the constant $\tilde{D}_{\eta_0}(V)$ *is given by*

$$\tilde{D}_{\eta_0}(V) = \max_{x \in \tilde{\mathcal{S}} \setminus \mathcal{S}} \min_{y \in \mathcal{S}} \frac{H(x,y) - U(x)}{U(x) - U_{\min} + \eta_0} < D(V).$$

COROLLARY 3.2.

$$\limsup_{N \to +\infty} (\log N)^{-2} \log \inf_{T,M,\eta_0,\rho} P(U(X_N) > U_{\min} \mid X_0 = x) \leq -\frac{1}{4\ \log(1+D)}.$$

The interest of this theorem lies mainly in its corollary, which shows that a proper tuning of the parameters leads to a faster scale of convergence speed than the one achieved by simulated annealing (see [7]). This remark of course deals with the comparison of two long runs of both algorithms. For repeated trials of bounded length, which we will consider in section 6.3, the question of knowing which algorithm is faster is open.

We will discuss practical means of choosing the parameters in connection with the jigsaw puzzle benchmark.

4. Solving jigsaw puzzles. We will illustrate on jigsaw puzzles the different steps of the general method of resolution.

First of all, we have to choose a search space. This will be the set of partial solutions where only some of the pieces are put in the frame.

$$\tilde{\mathcal{S}} = \{x \subset B \times E : x(r) = [a_r, a_r + w_r[\times[b_r, b_r + h_r[\,, \\ r \in \pi_B(x), x(r) \cap x(r') = \emptyset, r \neq r' \in \pi_B(x)\}.$$

Let us define now $q_C(x,.)$, the constructive dynamic starting from state x:

- First choose $r \in B \setminus \pi_B(x)$ according to the uniform distribution on this set.
- Then choose $(z,t) \in E \setminus \pi_E(x)$ according to the uniform distribution.
- Then try to expend this germ to a rectangle $[a_r, a_r + w_r[\times[b_r, b_r + h_r[$ of the desired size by adding alternatively a column to the left (or else to the right) and a line to the top (or else to the bottom). If it is not possible to grow the germ to its final size, just abandon the construction.
- Then draw a number k at random in the interval $[0, \text{max\_drift}[$ and move the location of $[a_r, b_r[$ k steps along the direction $(-1,-1)$ (that is, to the upper left corner, according to usual image indexing) if there is enough room to do so, or else move it as far as possible in this direction (until it bumps into other pieces).

The last two actions are better described by the following self-explanatory pseudo-C code, where $[a, c\,[\times[b, d\,[$ is the current germ:

```
int expend() {
  a=z; c=z+1; b=t; d=t+1;
  while((test1=(c-a<w))||(test2=(d-b<h))) {
  if (test1&&grow_left()&&grow_right()) return 1;
  if (test2&&grow_up()&&grow_down()) return 1;
  }
  for (k=rand(0,max_drift);k;k--) {
  if (move_left()&move_up()) break;
  } return 0;
}
```

where the functions `expend()`, `grow_left()`, `grow_right()`, `grow_up()`, `grow_down()`, `move_left()`, and `move_up()` return 0 on success and 1 on failure.

The destructive dynamic q_D is simpler:

- Draw $r \in \pi_B(x)$ at random,
- Form $y = x \cap \pi_B^{-1}(B \setminus \{r\})$, the partial solution where the piece labeled "r" has been removed from the frame.

The mechanism that was chosen for the constructions is meant to discourage the formation of small gaps between pieces. If nothing were done, when the discretization step of the grid is fine, small gaps would be left between the pieces with a large probability, and a complete solution to the puzzle, where pieces necessarily stick together, would never be discovered.

We have now to choose an energy function. Here again we will discourage the formation of gaps between pieces by introducing a term proportional to the contact length. By contact length we mean the sum of the contact lengths between pieces and between pieces and the edge of the frame.

Let μ be the counting measure on E. We take

$$U(x) = -\mu(\pi_E(x)) - \alpha \times \text{contact-length}.$$

For this choice of U, we can take the constant γ in equation (2) to be equal to the size of the smallest piece:

$$\gamma = \min_{r \in B} w_r \, h_r.$$

5. Minimizing a loss function.

5.1. Statement of the problem. We will discuss in the next two sections the case where some loss function $V : \mathcal{S} \longrightarrow \mathbb{R}$ has to be minimized on the state space $\mathcal{S}$ of global solutions of a task assignment problem. We consider the same framework as in the first section, with the difference that the problem is now to find a solution x belonging to $\arg\min_{y \in \mathcal{S}} V(y)$.

5.2. A general method of resolution. We will extend the method of section 3 to deal with a loss function.

The two first steps, building the search space and the constructive and destructive dynamics, will be the same as in section 3.

The change comes from the choice of the energy function. First we need to extend the loss function V to the search space $\tilde{\mathcal{S}}$ of partial solutions. Ideally, we would like to use the extension $\overline{V} : \tilde{\mathcal{S}} \longrightarrow \mathbb{R}$ defined by

$$\overline{V}(x) = \min\{V(y) \,:\, y \in \mathcal{S}, x \subset y\}.$$

Usually this is not an easily computable function, but in many situations there is a natural way to define a loss function for partial solutions. A simple way to do so, if there is nothing else at hand, is to set $V(y) = c$ for $y \in \tilde{\mathcal{S}} \setminus \mathcal{S}$, where c is a constant and $c \geq \max_{x \in \mathcal{S}} V(x)$. Then we build a compound energy function

$$W(x) = \alpha \, U(x) + V(x), \qquad x \in \tilde{\mathcal{S}},$$

where the real positive coefficient α is chosen such that, for some positive constant γ,

$$\begin{cases} \arg\min\limits_{x \in \tilde{\mathcal{S}}} W(x) \subset \mathcal{S}, \\ W(y) - W(x) \leq -\gamma < 0, \quad x \subset y, \ x \neq y \in \tilde{\mathcal{S}}. \end{cases}$$

These conditions are always satisfied for α large enough. However, the difficulty D of the energy landscape, related to the performance of simulated annealing, tends to $+\infty$ when α tends to $+\infty$. Therefore it is better to keep α as small as possible. In the next section, we will give an example for which we can take α arbitrarily small, and even $\alpha = 0$ if we are satisfied with $\gamma = 0$.

Equipped with this new energy function, we can proceed just as in the simpler case of section 3.

5.3. Some example of useful loss function. Often in task assignment problems, we would like some resources to be distributed according to some prescribed distribution. For instance, in a time-table problem we may want to schedule an equal number of hours in each week of the year.

This can be formalized in the following way. We consider first some function $\Phi : B \times E \longrightarrow F$, where F is a finite set (which may be the discretization of a domain in $\mathbb{R}^n$). Typically, Φ will be the projection on the time axis in a time-table problem. Then we consider a target distribution ρ defined on F. Let us consider some reference measure μ on $B \times E$ (such as the counting measure). To each partial solution $x \subset B \times E$, we may associate the restriction μ_x of μ to x, defined by

$$\mu_x(A) = \mu(x \cap A).$$

This induces a measure $\mu_x \circ \Phi^{-1}$ on F. The constraint we would like to represent by a loss function is that $\mu_x \circ \Phi^{-1}$ is approximately proportional to the reference measure ρ. This can be reflected in a loss function of the type

$$V(x) = \int h\left(\frac{\mu_x \circ \Phi^{-1}}{\rho}\right) d\rho,$$

where $h(x) = (1-x)^2$ or $h(x) = 1-x+x\log x$. The function h is in both cases strictly convex, satisfies $h(1) = h'(1) = 0$, and h' is strictly increasing; therefore $\mu_x \circ \Phi^{-1} = \rho$ if and only if $V(x) = 0$ and the minimum of $V(x)$ on the set $\mu_x(B \times E) = \textit{constant}$ is attained when $\mu_x \circ \Phi^{-1}$ is proportional to ρ, when this is feasible.

The following proposition holds.

PROPOSITION 5.1. *Assume that the total weight $\mu_x(B \times E)$ of any solution is a function of the tasks to be scheduled only. This means that there is a measure $\tilde{\mu}$ on B such that*

$$\mu_x(B \times E) = \mu(x) = \tilde{\mu}(\pi_B(x)), \quad x \in \tilde{\mathcal{S}}.$$

Then for all global solutions $x \in \mathcal{S}$, $\mu_x(B \times E) = \mu(x) = \tilde{\mu}(B)$ is a constant.

Assume moreover that the measure ρ defining the constraint is such that $\rho(F) \geq \tilde{\mu}(B)$, and assume also that

$$\left\{ x \in \mathcal{S} \,:\, \frac{\mu_x \circ \Phi^{-1}}{\rho} \equiv \textit{constant} \right\} \neq \emptyset.$$

Then

$$\arg\min_{s \in \tilde{\mathcal{S}}} V(x) = \left\{ s \in \mathcal{S} \,:\, \frac{\mu_x \circ \Phi^{-1}}{\rho} \equiv \textit{constant} \right\},$$

meaning that the partial solutions minimizing V are exactly the global solutions x for which $\mu_x \circ \Phi^{-1}$ is proportional to the constraint ρ.

The assumptions of the proposition will be satisfied when $\mu_x(B\times E)$ measures the amount of assigned resources, and the amount of resources to be allocated to a task depends only on the task and not on the way it is scheduled. Typically, for instance, the number of hours of a course of teaching will be prescribed in advance and will not depend on the choice of a schedule for the lectures.

Now let us make the supplementary assumption that $(\mu_x \circ \Phi^{-1}/\rho) \leq 1$ for any $x \in \tilde{\mathcal{S}}$. We can always make this assumption true by increasing ρ by a suitable multiplicative factor (at least when ρ is strictly positive on F). In some cases we may, on the contrary, want to restrict $\tilde{\mathcal{S}}$ by adding the new constraint $(\mu_x \circ \Phi^{-1}/\rho) \leq 1$. This will be done when the constraint has a practical meaning for the problem. For instance, if $\mu_x \circ \Phi^{-1}$ measures the number of lectures taking place in each hour of time in the week, we may want to fix ρ to a constant equal to the total number of available lecture rooms, add the constraint $(\mu_x \circ \Phi^{-1}/\rho) \leq 1$ to indicate that there is to be enough rooms to schedule all the lectures, and use the loss function $\int h(\mu_x \circ \Phi^{-1}/\rho)d\rho$ to indicate that we would like the rooms to be evenly occupied during the week (in a weekly time-table problem).

If the assumption $(\mu_x \circ \Phi^{-1}/\rho) \leq 1$, $x \in \tilde{\mathcal{S}}$ holds, then only the decreasing part of h is used, and the loss function V is always increasing during a destruction and decreasing during a construction. Therefore if γ is the constant corresponding to U in Eq. (2), we will have

$$W(y) \geq W(x) + \alpha\,\gamma, \quad x, y \in \tilde{\mathcal{S}}, \quad x \neq y, q_D(x,y) > 0.$$

6. The practical issue of the choice of parameters. In practical situations, the critical constants of the energy landscape are usually unknown. Therefore it is not possible to rely on the theoretical results we recalled in preceding sections to choose the parameters of algorithms. In the following subsections, we explain how we set the parameters in the experiments about jigsaw puzzles.

6.1. Simulated annealing. The cooling schedule can be written as

$$\frac{1}{T_n^N} = \beta_{\min}\left(\frac{\beta_{\max}}{\beta_{\min}}\right)^{n/N}.$$

We choose $\beta_{\min}$ and $\beta_{\max}$ by looking at the repartition function of the energies of the explored states in simulations at constant temperatures. We keep a value of $\beta_{\min}$ for which the slope of the repartition function stays large up to the largest values of the energies, meaning that states with high energies have a significant probability to be explored. For $\beta_{\max}$ we require, on the contrary, a repartition function concentrated on the lowest energy values.

The theory tells us that we can safely underestimate $\beta_{\min}$ and overestimate $\beta_{\max}$, which makes their choice possible from a qualitative inspection of repartition functions.

Figures 6.1 and 6.2 are two examples of repartition functions, corresponding to values of $\beta_{\min}$ and $\beta_{\max}$ which have been retained during the experiments.

6.2. The iterated energy transformation method. In this case, the choice of parameters is perhaps less straightforward. The analogy with simulated annealing can serve as a guideline: the high temperature regime corresponds to the case $\tau = \tau_1$ (i.e., to the first energy transform used). The low temperature regime corresponds to

$$\tau = (1+\rho)\eta_0 - U_{\min}.$$

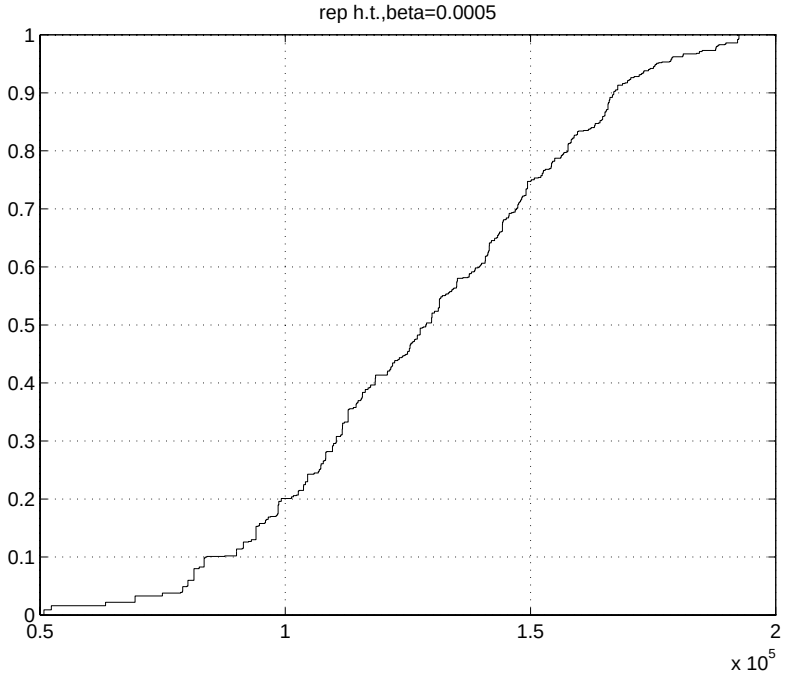
rep h.t.,beta=0.0005
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.5
1
1.5
2
x 10^5

FIG. 6.1.

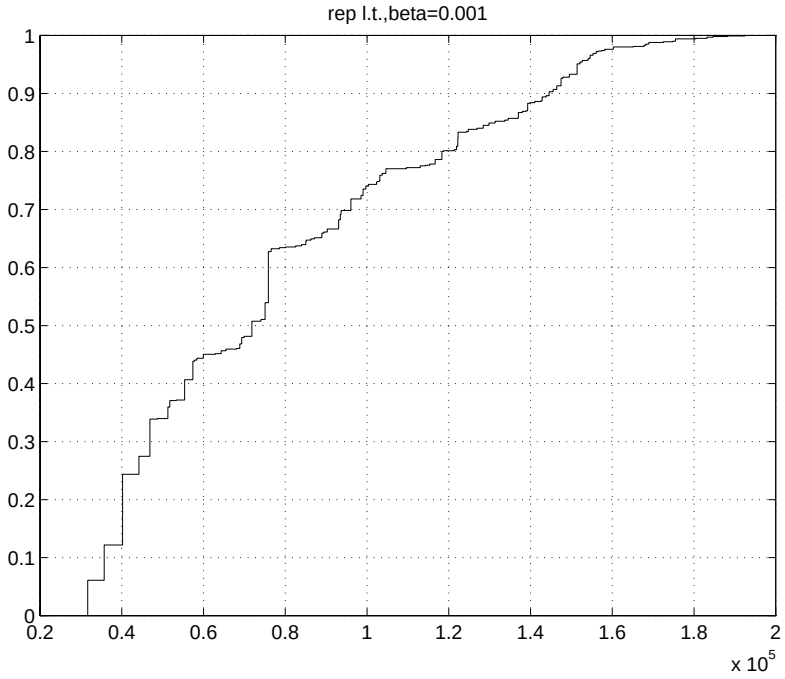
rep l.t.,beta=0.001
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
x 10^5

FIG. 6.2.

In order to test the behavior of the algorithm in these two configurations, we make a short test using a small value of ρ ($\rho << 1$). The law of evolution of τ_k shows that, for a small value of ρ, the algorithm will quickly switch from the high temperature regime during the first step to a low temperature regime during the following steps. In fact the value $\rho = 0$ may sometimes even be used. However, when this is done the algorithm sometimes encounters a state with a nondefined energy transform too quickly, and there are not enough iterations to compute a reliable repartition function for the low temperature regime. This problem, when encountered, can be circumvented by using a low but nonzero value of ρ.

We compute the repartition function of energies during the first step of the test run and during the last. The first function describes the equivalent of the "high temperature regime" and is tuned by the choice of the constant α and of the temperature parameter T; the second function corresponds to the "low temperature regime" and is tuned by a proper choice of $\tilde{\eta_0} = \eta_0(1+\rho)$.

Once these two choices are made, there remains a free parameter, namely, ρ.

The theory [7] indicates an optimal choice of ρ of order $\sqrt{N}$ and an optimal choice of $r = N/M$ of order $\sqrt{N}\log(N)$. On the other hand, as soon as $\log(1+\rho) > 1$, the convergence rate will be better than for simulated annealing. This indicates that a large value of ρ may safely be chosen and that r can then be set to make

$$(U_{\min} - \gamma + \eta_0)\rho\left(\frac{\rho}{1+\rho}\right)^{(r-1)}$$

small. This will ensure a small dependence of the final value of the shift τ_N with respect to its initial value $\tau_1 = \delta - \eta_0$.

6.3. Repeated optimizations. In this section, we will consider that N iterations are to be divided into N/M trials of length M, and that we will keep the best solution found out of these N/M trials. In this context, the probability of failure in the worst case with respect to the starting point of each trial is $\epsilon_1(M)^{N/M}$, where

$$\epsilon_1(M) = \max_{x\in\tilde{S}} P\left(U(X_N) > U_{\min} \mid X_0 = x\right).$$

The first remark to be made (see Azencott [2], [3]) is that for all the algorithms we have considered, $\lim_{M\to+\infty}(1/M)\log\epsilon_1(M) = 0$. Therefore when N is large enough, the optimal value for M is independent of N.

We will discuss here the choice of the length M of each run of the algorithm. For simulated annealing, we can, on the basis of the theoretical bound on the probability of failure, namely, $(A/M^\alpha)^{N/M}$ for N iterations divided into N/M runs of length M, conjecture that an overestimation of M will be relatively harmless, whereas an underestimation would be more penalizing. This can be seen on the derivative

$$\frac{\partial}{\partial M}\left(\frac{A}{M^\alpha}\right)^{N/M} = \left(\frac{A}{M^\alpha}\right)^{N/M}\frac{N\left(\alpha(\log M - 1) - \log A\right)}{M^2},$$

but is may be more vividly illustrated by a small numerical application. If we take, for example, $A = e^4$, $\alpha = 1$, and $N = 1000$, and if we put $\epsilon(M) = (A/M^\alpha)^{N/M}$, we see that

$$\min_M \epsilon(M) = \epsilon(148) \simeq 0.0012,$$

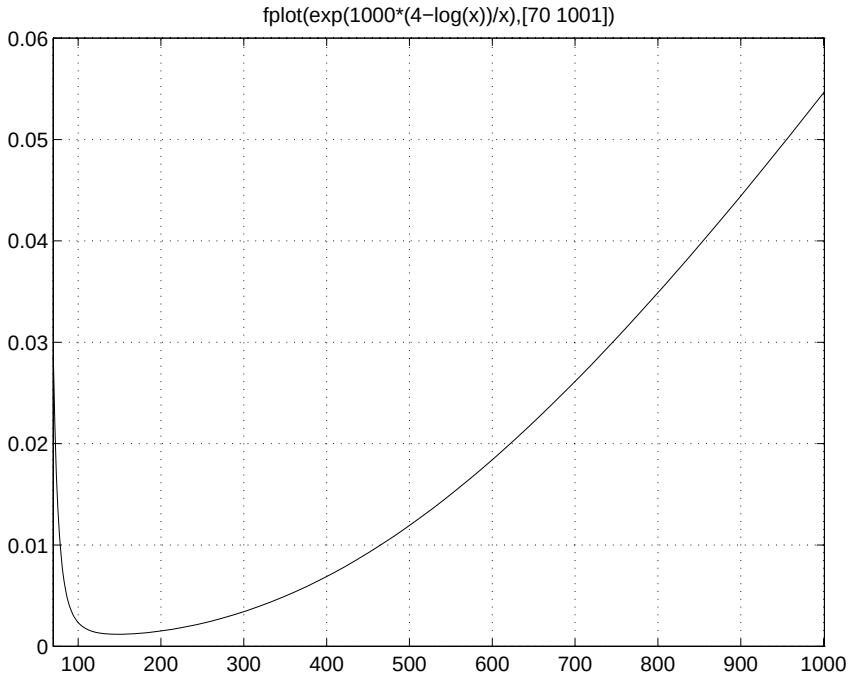

FIG. 6.3.

and that for this optimal value the probability of failure in each run is $\simeq 0.37$.

Here are some values taken by $\epsilon(M)$

M	74	100	148	300	500	1000
$\epsilon(M)$	0.016	0.0024	0.0012	0.0034	0.012	0.055

and a graphic of this function is shown in Fig. 6.3.

These figures show that, as far as this rough theoretical bound is a good guideline, there is a clear benefit in performing multiple runs instead of one long run, but that an overestimation of a factor two of the length of each run is relatively harmless. We remark also that a quite low confidence level for each run is favorable in this example where the difficulty is one.

The same kind of reasoning would also hold for the theoretical bound of order $\epsilon(M) = (A/M^{\alpha \log M})^{N/M}$ obtained for the iterated energy transformation method. In this case the derivative of the confidence level $\epsilon(M)$ is

$$\frac{\partial}{\partial M}\epsilon(M) = \epsilon(M)\frac{N\left(\alpha((\log M)^2 - 2\log M) - \log A\right)}{M^2}.$$

Figure 6.4 shows a plot of this function for some choice of the parameters α, N, and A: the tolerance with respect to an overestimation of M is even better than for simulated annealing.

For a comparison between repeated searches and interacting parallel searches, we refer to Graffigne [16] and Azencott and Graffigne [4].

6.4. A partial freezing method. In [7] we saw that the simulated annealing algorithm is not efficient to deal with a state space made of a large number of independent components. By "independent components," we mean the case when the energy

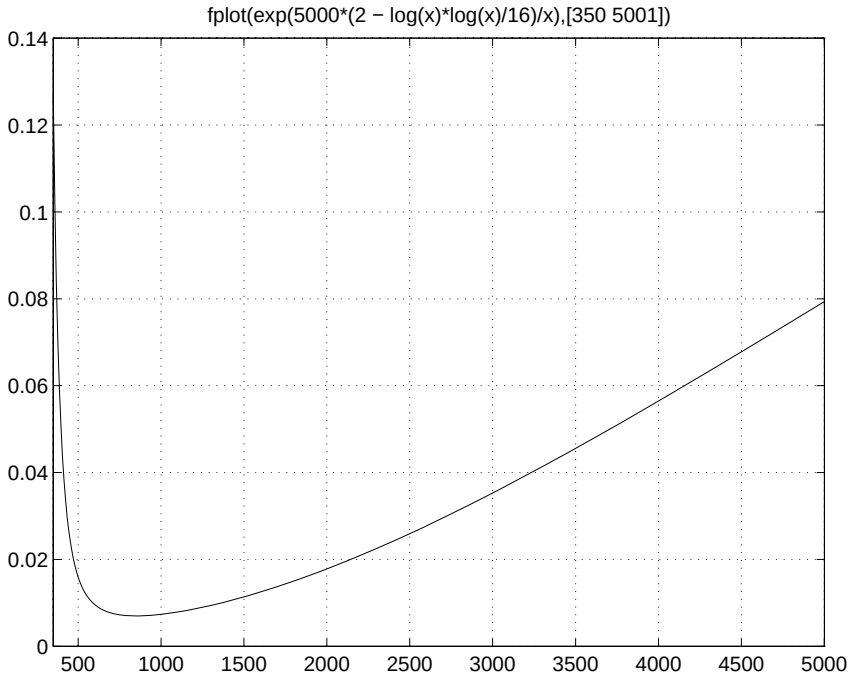

FIG. 6.4.

function is a sum of terms depending on distinct coordinates. Here the different tasks to be scheduled interact through the constraints and through the contact length term in the energy; therefore we cannot describe their assignment by independent coordinates. Nevertheless, in the end of the optimization process, we would like to be able to perform some last small local improvements on the current solution depending on distinct small subsets of tasks, with the assignment of the other tasks remaining untouched. We can write (formally) the energy function in a suitable neighborhood of the current solution as a sum of such possible local improvements. In the end of the optimization, we will be working at low temperature; therefore the current solution will, most of the time, be a local minimum and we will typically not try more than one local improvement at a time. Therefore (this is only a heuristic reasoning) we can expect the optimization process to behave approximately the same as in the independent component case when it draws to the end (that is, at low temperatures). It is easy to show (see [7]) that an efficient way to deal with independent components is to perform a series of local optimizations, resetting after each step the current solution to the best solution found. This will do much better than the global algorithms we described so far when the number of components is large. The reason is that a global algorithm cannot efficiently move one task around without disturbing the others. These considerations suggest adding a postprocessing to global optimization, made up of a series of local optimizations. When we put these things and the use of repeated optimizations together, we end up with a partial freezing method, which can be symbolically described by the following nested loops:

```
repeat
  reset the current solution to the empty assignment
  for (n taking increasing values from 0 to max)
    repeat
      choose at random a set of n ‘‘frozen’’ tasks among
      the scheduled tasks
      repeat
        run a stochastic optimization algorithm during
        which the frozen tasks stay untouched
      endrepeat
      reset the current solution to the best solution
      encountered in the previous loop
    endrepeat
  endfor
endrepeat
return the best solution encountered in the outer loop.
```

The stochastic algorithm used in the inner loop may be one of the three algorithms we studied here. It is applied to the subset of the state space defined by the current assignment of the frozen tasks. When the number of currently scheduled tasks is less than n, we freeze all the scheduled tasks. In the experimental section we will show results obtained with this partial freezing method applied to the iterated energy transformation algorithm. One advantage of the partial freezing method is that it is less demanding on the global optimization step and is therefore tolerant of a looser choice of the parameters of the algorithm.

7. Experimental results. We tried to solve two kinds of puzzles: a small "tight" puzzle with nine pieces and no loss function, and a big "loose" 60-piece puzzle with a loss function. By "tight" we mean that there is just enough room in the frame to put all the pieces, and by "loose" we mean, on the contrary, that there is some extra room left in the frame, the difficulty being then to minimize the loss function.

7.1. Small "tight" jigsaw puzzle. Our small jigsaw puzzle is a nine-piece problem. The algorithm we used to solve it corresponds to the description given in section 4. The frame is a 40×50 grid. The size of the pieces are $(14, 27)$, $(8, 36)$, $(8, 9)$, $(6, 14)$, $(34, 5)$, $(18, 9)$, $(22, 9)$, $(18, 21)$, $(18, 15)$. The problem has several solutions, due to symmetry properties. Figure 7.1 shows one solution: the parameters of the algorithms were set using the heuristics described in section 6.

We performed 40 runs of the simulated annealing algorithm and the same number of runs of the IET algorithm. For each algorithm, we computed the repartition function of the energy of the best solution encountered during each run and computed the repartition function of the energy of the final state of the algorithm. Of course, the former repartition function is always above the latter; therefore we can unambiguously plot them on the same diagram. In order to perform a "fair" comparison, we allowed the same number of iterations in both cases, namely, $N = 5000$ iterations per run.

The results (Figs. 7.2 and 7.3) are of the same order, with some advantage in favor of the IET method. This is especially true when the energy of the final state is considered. An interpretation of this fact is that the IET algorithm is more efficient in preventing the process from leaving the global minimum once it has reached it.

We were also able to check the influence of the drift towards the upper left corner.

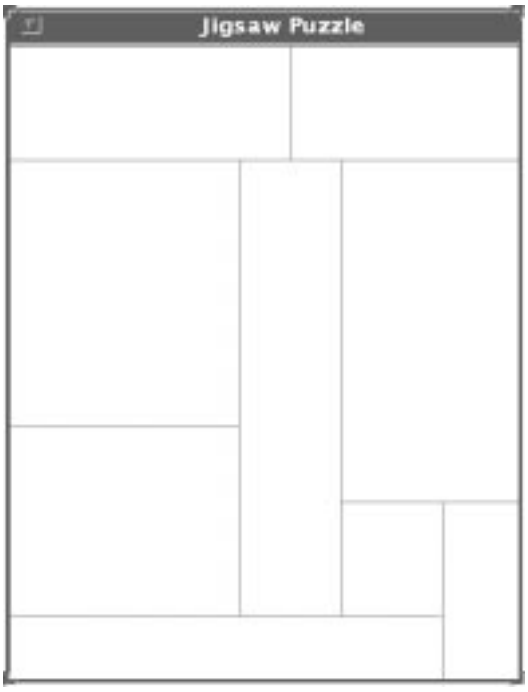

FIG. 7.1.

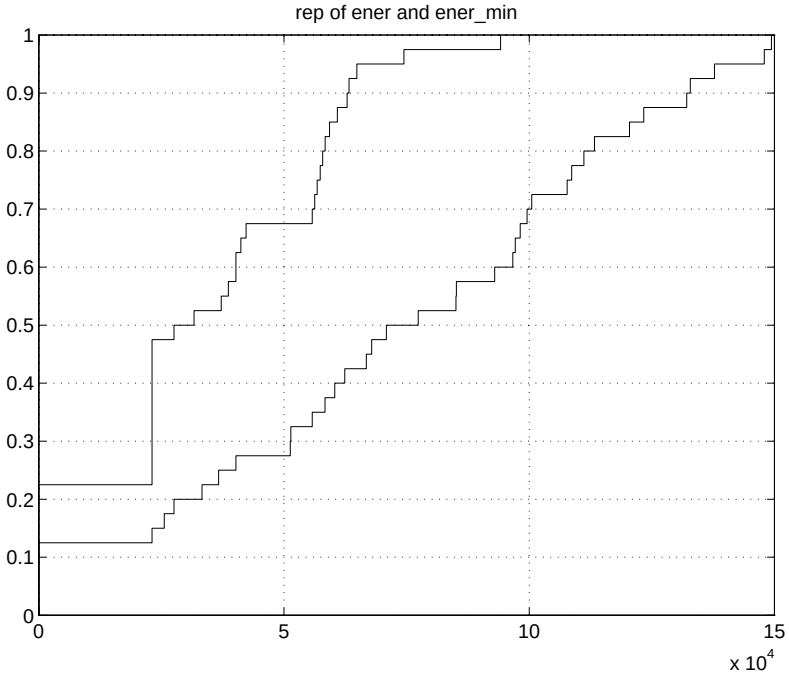

FIG. 7.2. *Performance of simulated annealing.*

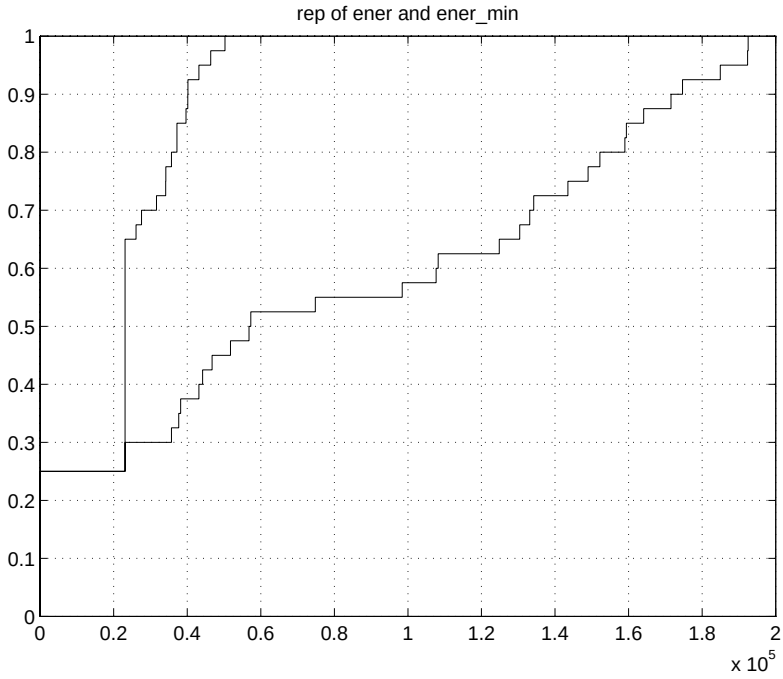

FIG. 7.3. *Performance of the iterated energy transformation method.*

In the two previous experiments, the maximum number of steps of the drift (the constant `max_drift` in the pseudocode of section 4) is 10. We have also tried a maximum number of steps of 50, for simulated annealing. We obtained on 40 runs the improvement in the performance shown in Fig. 7.4.

7.2. A big "loose" jigsaw puzzle. Our big jigsaw puzzle has 60 pieces, covering an area of 230 unit squares. The frame is a grid of size 30×10. The sizes of the pieces are the following:

number of pieces	width	height
15	3	2
15	2	1
5	5	2
5	2	4
20	1	1

The loss function is of the type described in the previous section. The function Φ here is the projection on the second axis, $\Phi((r, a, b)) = b$, $(r, a, b) \in B \times E$, so that the constraint indicates how much of each line the pieces should fill. On the following diagram, we have plotted the constraint function ρ (see Fig. 7.5).

With this choice of ρ, the constraint is tight, meaning that $\rho(F)$ is equal to the area of the pieces. When we use tight constraints, we build problems of the partition type, which are therefore NP complete. We chose the size of the pieces such that the set of global solutions is not empty. However, for a 60-piece problem, it is very difficult to find a (complete) solution.

We have chosen a coarse discretization step to keep the difficulty of the problem

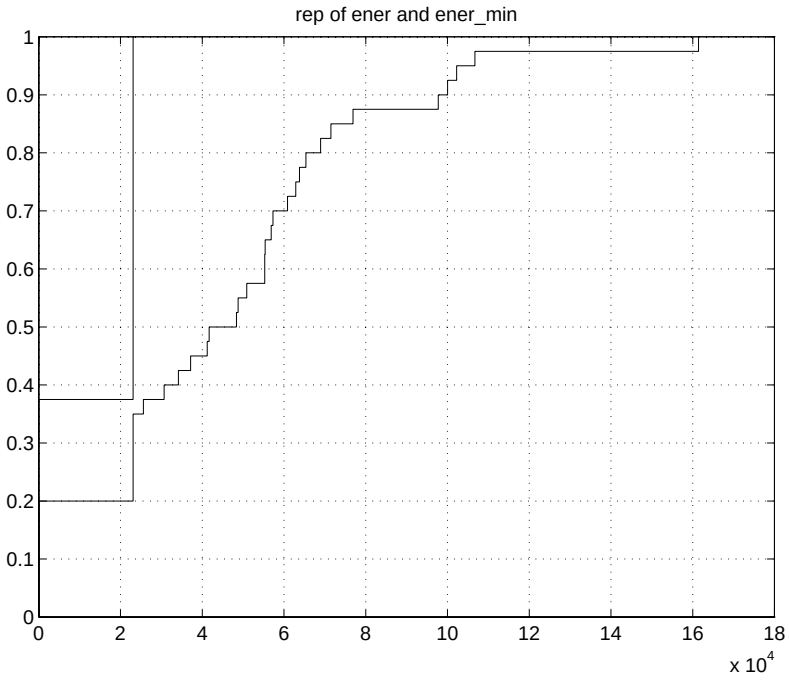
rep of ener and ener_min
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
2
4
6
8
10
12
14
16
18
x 10^4

FIG. 7.4.

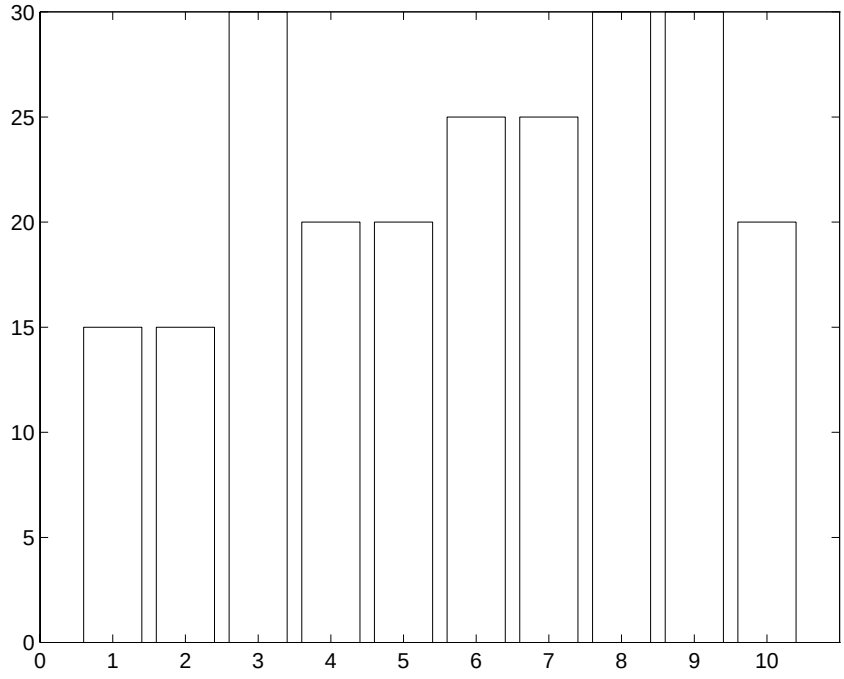
30
25
20
15
10
5
0
0
1
2
3
4
5
6
7
8
9
10

FIG. 7.5.

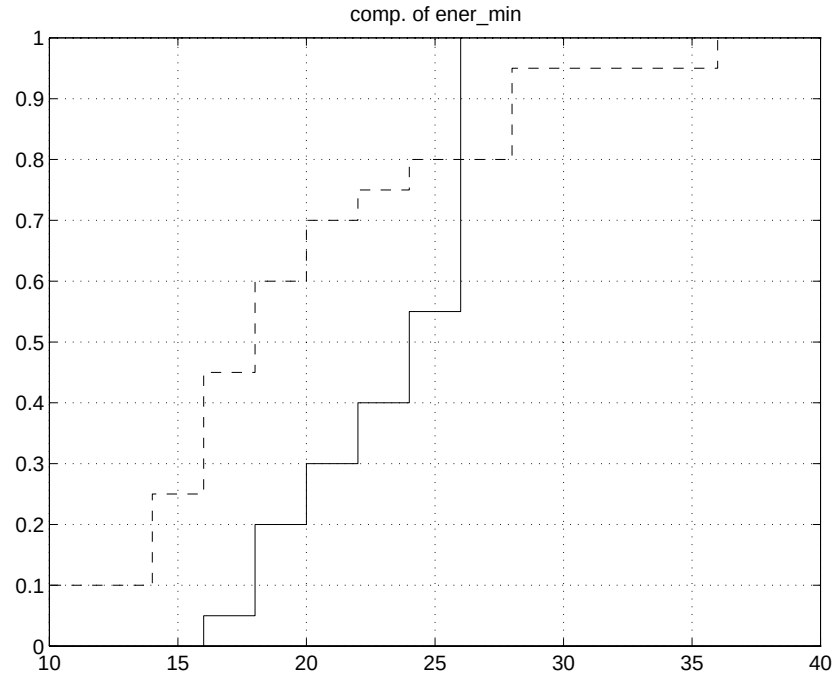

FIG. 7.6.

to a reasonable level, since we had to switch off the vertical drift. Indeed keeping a vertical drift would have decreased the stability of minimizing configurations in an unfavorable way.

We tried two kinds of energies. In order to have a point of comparison, we tried to use the simple energy $U(x) = -\mu(x) + \max_{y \in \tilde{\mathrm{S}}} \mu(y)$, where μ is the counting measure.

Then we tried a compound energy $W(x) = U(x) + \alpha V(x)$ for a large value of α and for $V(x) = \int (1 - (\mu_x \circ \Phi/\rho)^2 d\rho$.

Eventually, we tried to relax the constraint, changing ρ to $\tilde{\rho} = 6/5 \times \rho$.

7.2.1. Experiments with a simple energy function. In order to have a point of comparison, we recorded first the performance of repeated relaxations. The relaxation algorithm we used corresponds to a choice of $\lambda = 0$, or equivalently to a choice of $\beta = +\infty$ in the Metropolis algorithm.

Then we considered the Metropolis algorithm for different values of λ and of β. We tried $\lambda = 1$ and $\lambda = 0.5$, two "natural" choices for λ. The former let us inhibit destructions only according to the energy increment, whereas the latter let constructions and destructions have equal frequencies at infinite temperature.

The first conclusion we reached was that a significant improvement over the relaxation scheme could be obtained using the Metropolis algorithm with a moderate number of steps. We compared relaxation with 300 steps (for which convergence was always reached) with Metropolis with $\lambda = 1$, $\beta = 1$, and $N = 4000$. In order to compare methods using the same number of iterations, we repeated Metropolis 20 times and the relaxation algorithm $\lfloor (4000 \times 20)/300 \rfloor = 266$ times. On the following diagram (Fig. 7.6) we plotted the repartition functions of the best solution found for each of the 20 runs of Metropolis (dashed lines), along with the best 20 results out of the 266 runs of the relaxation algorithm (solid lines).

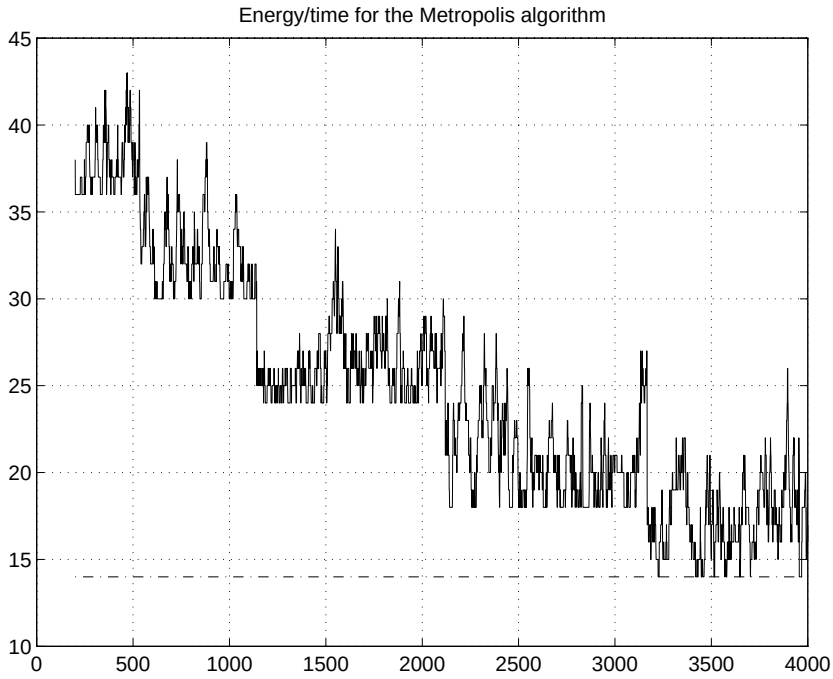

FIG. 7.7.

We obtained very suggestive evolutions for the Metropolis algorithm, such as the following (Fig. 7.7).

On this plot of $u_n = U(X_n)$ for $n = 300, \dots, 4000$, we see the "staircase" shape of the trajectories of the Metropolis algorithm. The algorithm "falls" into deeper and deeper maximal cycles [13] of the domain $\tilde{\mathcal{S}} \setminus \mathcal{S}$. We refer to [5] for a theoretical study of the exit path of the Metropolis algorithm from a domain at low temperature. For a study of the trajectories of simulated annealing algorithms, we refer to [6] and [22], which rely on more complex but also more general induction proofs which cover the time inhomogeneous case. For a semigroup approach of the same question in the continuous time case, we refer to [10], [11], [12], [17], [19], [20], and [21].

The energy evolution can be decomposed into a decreasing part $\underline{u}_n = \min_{k \le n} u_k$ and a "wandering" part $\overline{u}_n = u_n - \underline{u}_n$, as in the following diagram (Fig. 7.8).

The repartition function of the wandering part gives information about the depth of secondary attractors from which the algorithm is able to escape within the time of the simulation. It is a useful tool to choose the inverse temperature parameter β. Following is the repartition function corresponding to the preceding plot (Fig. 7.9).

The best results for the Metropolis algorithm of time length $N = 4000$ were obtained for $\beta = 1$ and $\lambda = 1$ or for $\beta = 0.8$ and $\lambda = 0.5$. This shows that in this case, the choice of λ is not crucial. In the following, we will use $\lambda = 1$, because we can hope to take better advantage of the discrimination made by the energy function between small and big pieces when we use this value of λ.

Then we used the Metropolis algorithm and simulated annealing on long time intervals. Namely, we took $N = 20000$, $\beta_{\min} = 0.7$, $\beta_{\max} = 1.1$ for simulated annealing and $\beta = 1$ for the Metropolis algorithm. On 10 runs of each algorithm, we could notice a clear gain in performance in favor of simulated annealing.

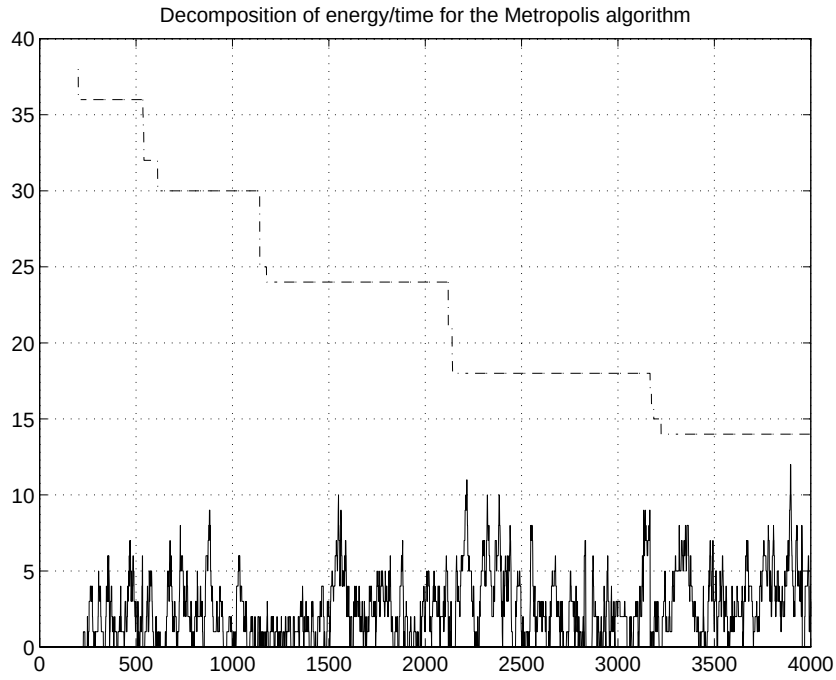
Decomposition of energy/time for the Metropolis algorithm
40
35
30
25
20
15
10
5
0
0
500
1000
1500
2000
2500
3000
3500
4000

FIG. 7.8.

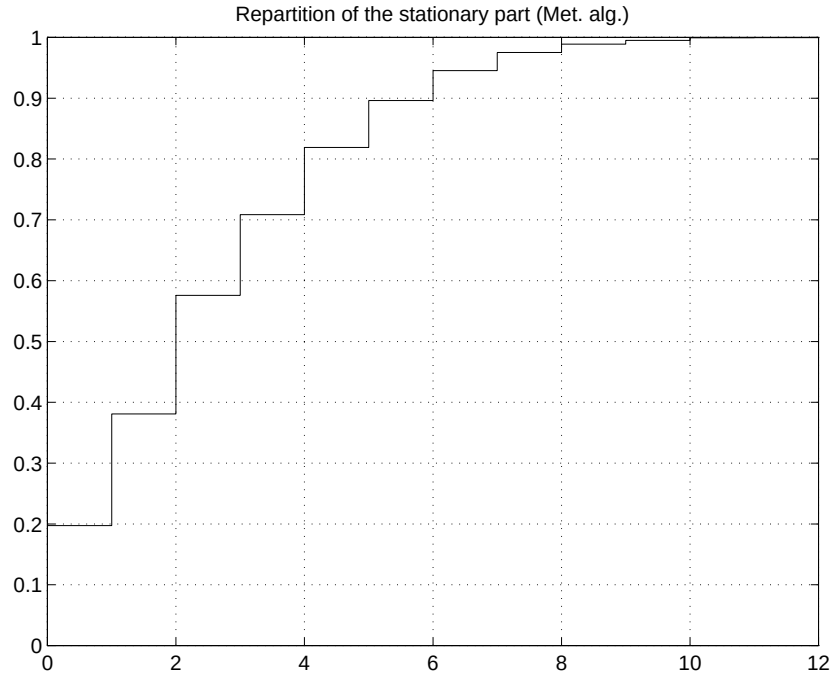
Repartition of the stationary part (Met. alg.)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
2
4
6
8
10
12

FIG. 7.9.

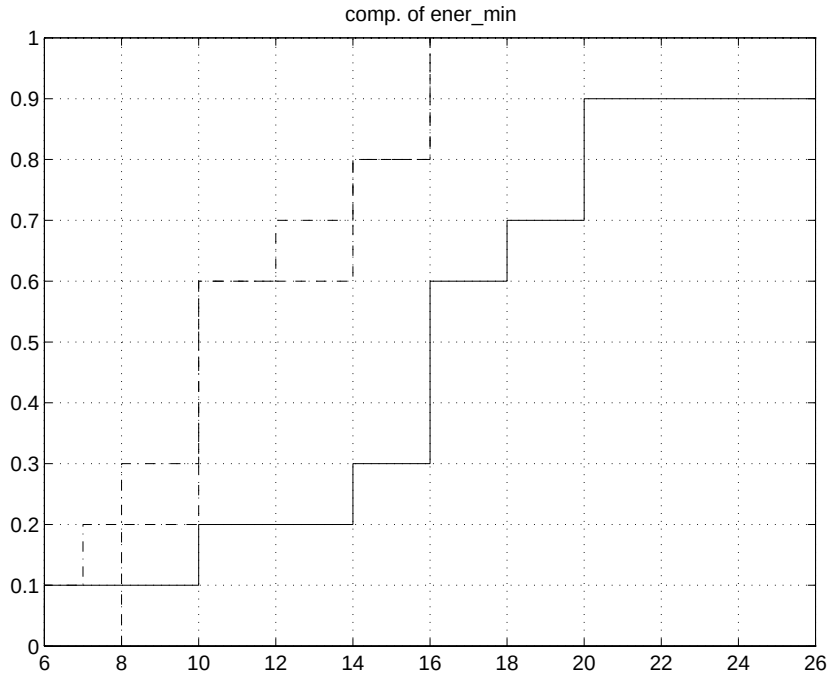

FIG. 7.10.

We tried eventually to get a better improvement using the IET algorithm. Since the state space is already rather large, we followed the idea introduced in [7] to use transformations $F_{\alpha,T,\tau}$ with a nonzero value of α.

We took $\alpha = 0.3$, $\beta = 1/T = 30$, $\frac{1}{1+\rho} = 0.5$, $r = 4$, and $\eta_0 = 15$. We obtained the following comparative results for the best energy found in each of 10 runs of each algorithm. The mean values are 16.2 for the Metropolis algorithm (solid lines), 11.6 for simulated annealing (dashed lines), and 10.9 for the IET algorithm (dash-dot lines). The repartition functions are plotted on the next diagram (Fig. 7.10).

7.2.2. Experiments with a compound energy function. We used the energy

$$W(x) = U(x) + \alpha V(x),$$

with a huge value of $\alpha = 10000$.

The range of this energy is very large, when compared with the previous one, since $W_{\max} = 2300230$, whereas $W_{\min} = 0$ and removing a piece of size 1×1 from a complete solution in a line of weight $\rho(y) = 30$ costs $\Delta W \simeq 334.33$. Therefore we may expect more spectacular improvements from the speed-up techniques.

We tried different temperatures for the Metropolis algorithm with $N = 20000$. The best results were obtained when $\beta = 8 \times 10^{-4}$. On 10 runs, the average best value was 15853.

Using simulated annealing with $\beta_{\min} = 10^{-4}$, $\beta_{\max} = 10^{-3}$, we improved the performance on the average, as shown in the next diagram. On 10 runs, the average best energy value was 8765.

We obtained some more improvement using the IET algorithm (with $\gamma = 5 \times 10^{-5}$, $\beta = 10$, and $\eta_0 = 2000$). On 10 runs the average best energy value was 6280.

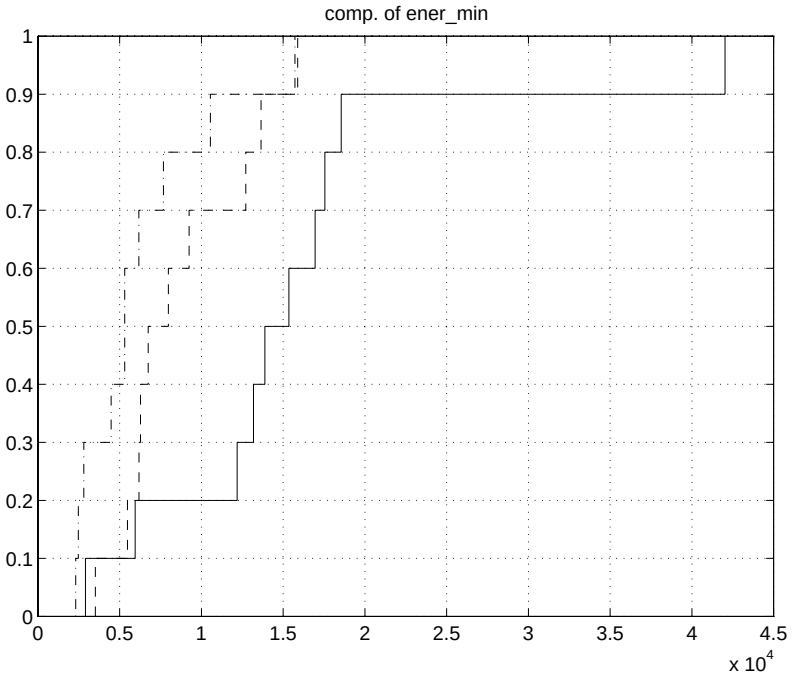

FIG. 7.11.

Figure 7.11 shows a diagram of the repartition functions of the best energy value for ten runs of the Metropolis algorithm (solid lines), simulated annealing (dashed lines), and the IET algorithm (dash-dot lines).

7.2.3. Experiments with a relaxed constraint. We explored also an alternative in the optimization design, which consists of replacing ρ by $\tilde{\rho} = \frac{6}{5}\rho$. We considered accordingly a larger search space $\tilde{\mathcal{S}}$ where the constraint $\frac{\mu_x \circ \Phi^{-1}}{\rho} \leq 1$ is relaxed to $\frac{\mu_x \circ \Phi^{-1}}{\tilde{\rho}} \leq 1$. We took again a compound energy of the type $W(x) = U(x) + \alpha\, V(x)$, with $\alpha = 10000$. The range of W is between $W_{\min} = 76666.66$ and $W_{\max} = 2760230$.

In this example, we can perform the same kind of comparison as in the case of tight constraints. We made 10 runs of length $N = 20000$ of each algorithm. The average of the best energy value found in each run is 8731 for the Metropolis algorithm, 8685 for simulated annealing, and 8567 for the IET algorithm. Figure 7.12 shows a diagram of the corresponding repartition functions (solid lines for the Metropolis algorithm, dashed lines for simulated annealing, and dash-dot lines for the IET algorithm).

The best solution was found by the IET algorithm. It has an energy of $W(x) = 78750$ and is shown in Fig. 7.13.

In this solution, all the pieces are set in the frame. We can judge the quality of the solution with respect to the proportionality constraint on the following diagram (Fig. 7.14), where we have plotted $\tilde{\rho}$ (dashed lines), the measure expressing the constraint, and $\mu_x \circ \Phi^{-1}$ (solid lines), giving the number of unit squares actually filled on each line by the solution. The optimum would be $\mu_x \circ \Phi^{-1} = \rho = 5/6 \times \tilde{\rho}$. We are not too far from that: the two entries $\mu_x \circ \Phi^{-1}(2)$ and $\mu_x \circ \Phi^{-1}(4)$ are one unit too large, and $\mu_x \circ \Phi^{-1}(9)$ is two units short from the optimum. This is the best approximation to an optimal solution we were able to compute on this example. This seems to show that

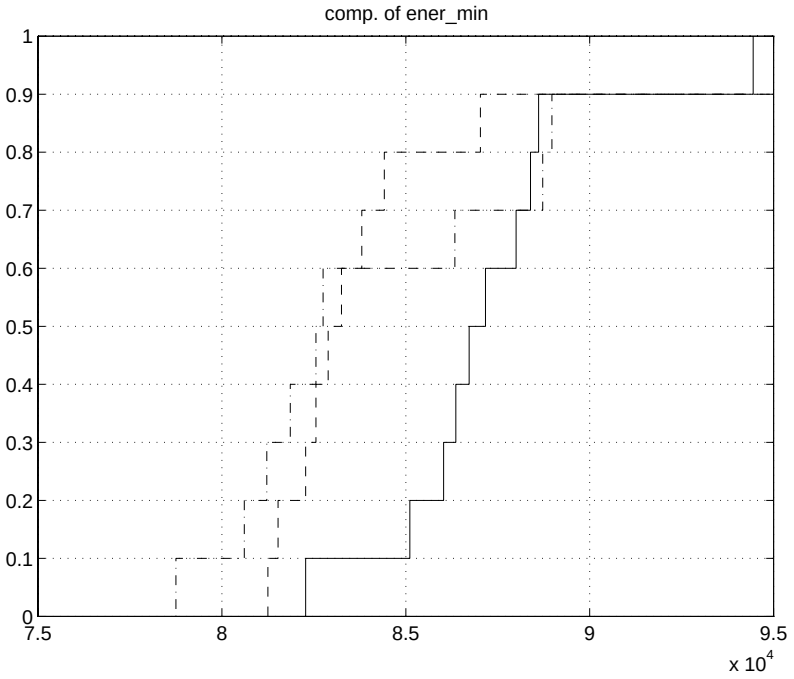

FIG. 7.12.

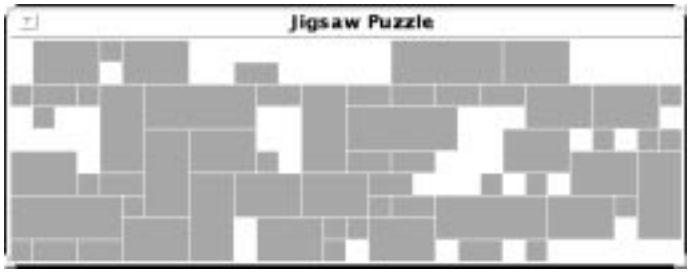

FIG. 7.13.

relaxing the constraint slightly and introducing the loss function $V(x)$ in the energy eases the optimization process.

This should be compared with the best solution found without relaxing the constraint (Fig. 7.15) and its constraint diagram (Fig. 7.16).

For this solution, $U(x) = 6$. Solutions of energy $U(x) = 6$ were also found using the simple energy U to guide the search. Therefore the advantage of introducing the V component in the energy function is not obvious when the constraint is really tight.

It is also interesting to consider typical energy evolutions of those three algorithms. On the following diagrams (Figs. 7.17–7.19), we have plotted the sequence

$$u_n = U(X_n).$$

As we have already mentioned, these sequences of energy values can be decomposed

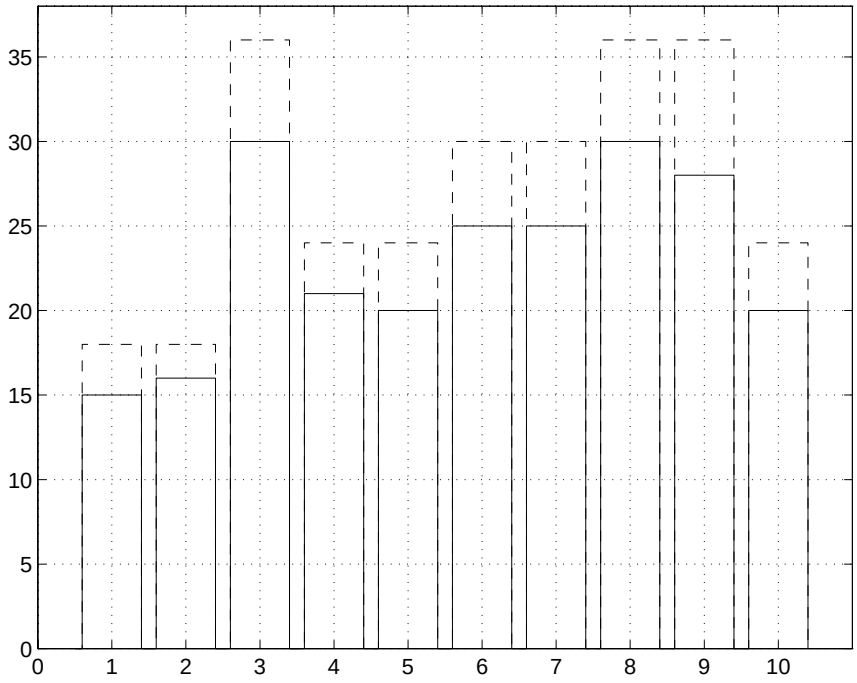

FIG. 7.14.

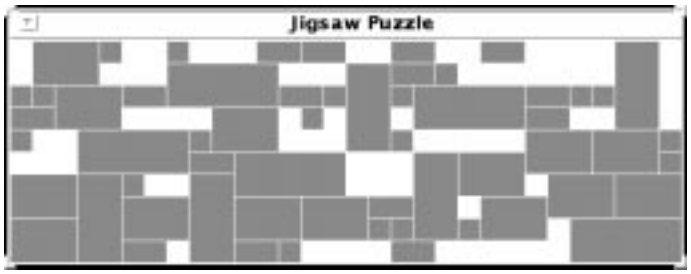

FIG. 7.15.

into a decreasing component

$$\underline{u}_n = \min\{u_k \,:\, k \leq n\},$$

and a wandering component

$$\overline{u}_n = u_n - \underline{u}_n.$$

The repartition functions of $(\overline{u}_n, n = 1, \ldots, N)$ can help to properly set the parameters. It indicates the depth of the attractors from which the algorithm is able to escape.

It is interesting to compare the energy evolutions of the three algorithms. The comparison between the Metropolis algorithm and simulated annealing shows clearly that the temperature used in Metropolis is too low during the first 4000 iterations and too high during the last 8000 iterations. As for the IET algorithm, we can see

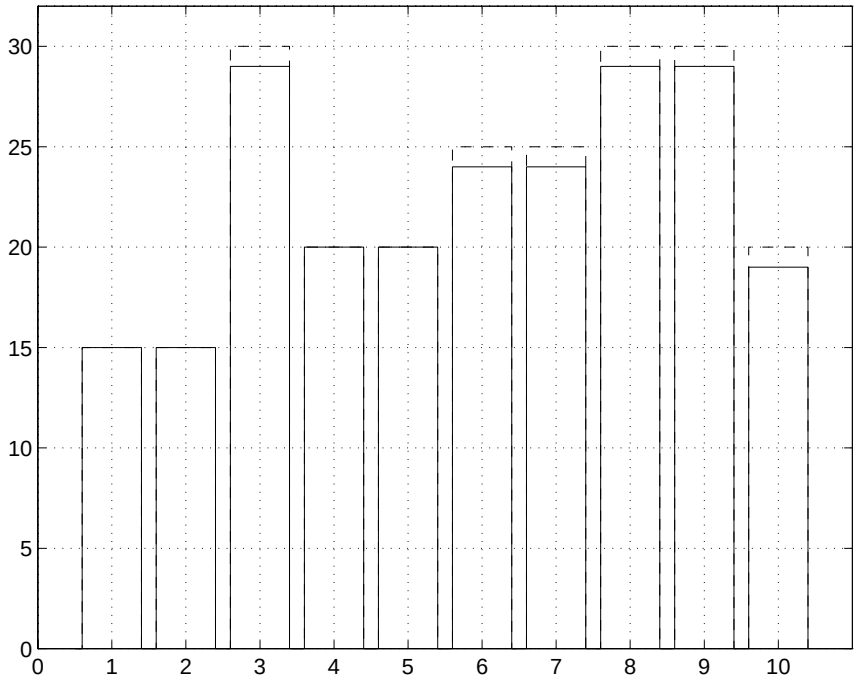

FIG. 7.16.

that the fluctuations of the wandering part are decreasing with time, as in the case of simulated annealing, but that the evolution of the energy is more unstable: it can go up and down faster (in other words, its peaks are sharper). This explains why it is able to sample more efficiently a state space containing many local minima.

7.2.4. Experiment with the partial freezing method. In this experiment, we took the IET algorithm, which had proved to be the best when used globally, and we added a postprocessing stage where we froze all but three of the tasks. At the same time we decreased the global optimization step from 20000 iterations to 3000 iterations and kept 50 times 300 iterations for postprocessing (we drew 50 different frozen configurations and made 300 iterations for each; in each frozen configuration, only three pieces were left unfrozen). Thus we decreased slightly the total number of iterations from 20000 to 18000 (in answer to a suggestion of one of our referees that a more complex algorithm should be allowed less iterations). At the same time, we got an improvement on 10 trials for both the mean value of the energy and its minimum value over the 10 trials (Fig. 7.20). We also found that it was much easier to tune the parameters of the IET algorithm.

Figure 7.21 shows an energy evolution typical of the partial freezing method, where the lower plot shows the evolution of $\eta_0 - \tau_n$: we see that the partial freezing allows us to go faster up and down the energy landscape (in other words, it allows us to work at a higher temperature).

7.2.5. Is the number of iterations a fair measure of complexity? In the case of the three algorithms in this paper, the inner loop is the same except for the computation of the rejection probabilities. When one goes from the Metropolis algorithm to simulated annealing, one has to add the cost of updating the temperature,

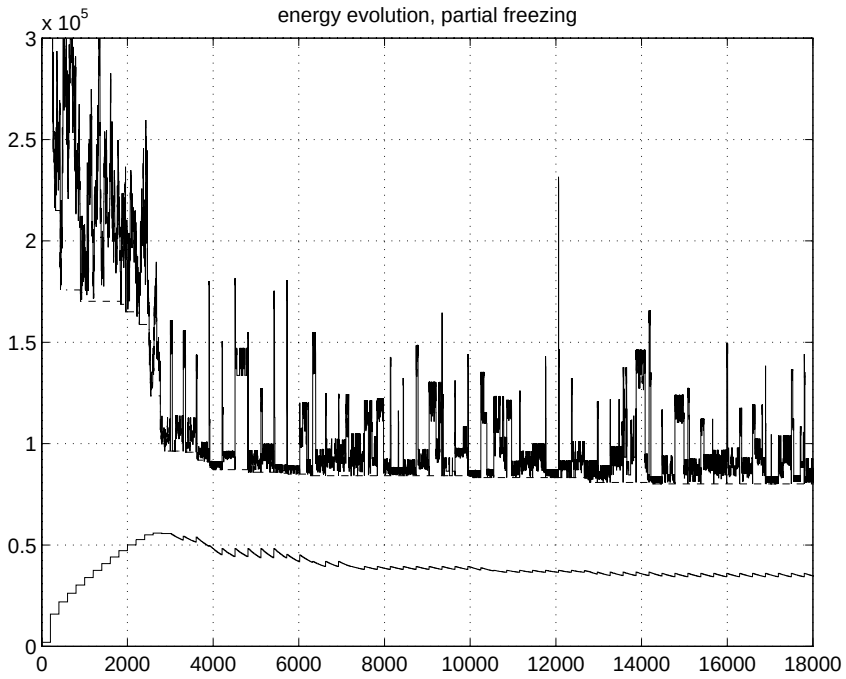

FIG. 7.17.

that is, the cost of one multiplication (in fact, it is even possible to use piecewise constant temperature schedules with the same theoretical properties; see [9], for which the update of the temperature is not in the inner loop and therefore has little influence on the computer time). When one goes from the Metropolis algorithm to the IET algorithm, one has to compute the energy transformation, that is, $\log \frac{U+\Delta U}{U}$, this means one addition, one multiplication, and a logarithm. The fact that one uses the value of U and not only the value of ΔU, the energy increment, does not really make a difference in practice since one will, anyhow, want to record the value of the energy U in order to keep the best solution encountered and not systematically keep the last current solution. Anyhow, computing U from the accumulated energy increments requires only one addition per iteration.

In fact, in our experiments, these differences in the computing time of the rejection probability does not seem to be the leading factor in the variations of the cpu time. The unix function "time" gave us the following figures: 66.3 seconds of user cpu time for 20000 iterations of the simulated annealing algorithm, 61.1 seconds for 20000 iterations of the IET algorithm, and 35.2 seconds for 18000 iterations of the IET algorithm combined with the partial freezing method. These figures are somewhat unexpected. Our interpretation is that all the moves do not have the same complexity. This is particularly true with the partial freezing method, where during most of the time (15000 iterations) the state space is restricted; therefore the choice of a task to schedule or to destroy and the choice of the resources to allocate are made from smaller sets and therefore are faster. To a minor degree the same happens with the IET algorithm and Simulated Annealing: the IET algorithm spends more time at low energy levels where more tasks are scheduled and where scheduling a new task is done from a smaller set of available tasks and resources.

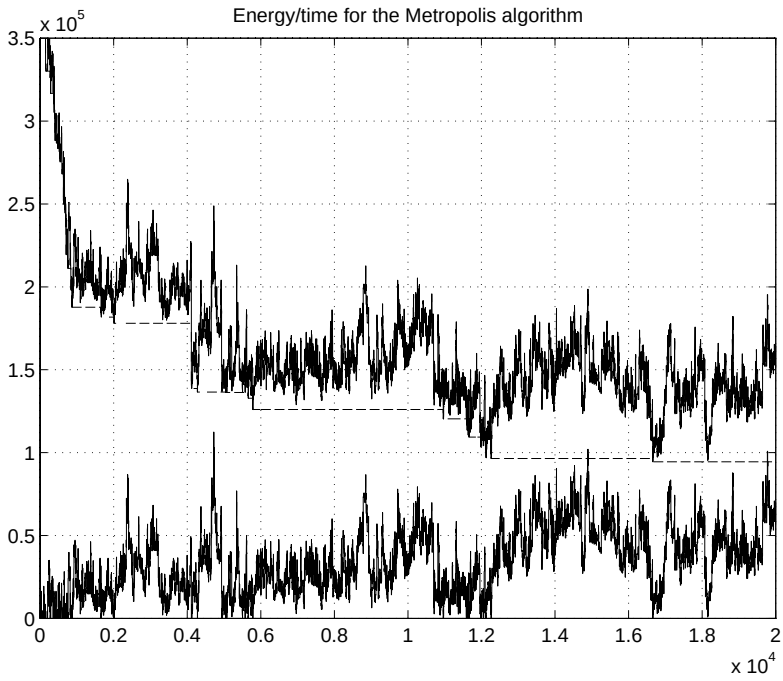

FIG. 7.18.

In conclusion, the number of iterations is not a perfect measure of complexity but has the advantage of being machine independent. To get a real cpu time complexity study, one would need to analyze dynamically the complexity of moves, which depends on the current configuration, and to gather statistics from profile files, which we have not done in the framework of this study.

8. Comparison with other energy landscapes. The most common way to enlarge the space of solutions to create a search space is to allow overlaps. In the task assignment formulation, this means that the same resource is allowed to be used by more than one task at the same time. In the jigsaw puzzle formulation, this means that we allow pieces to sit on top of each other. We will maintain the jigsaw puzzle terminology in the following discussion.

Let us discuss first the case where the aim is simply to find a complete admissible solution. We will discuss afterwards the case where a cost function has to be optimized on the set of complete solutions.

So for the moment, the energy in the overlap case will be made up of the total area of the overlaps and, in the partial solution case, will be made up of the total area of unused pieces.

We can expect the overlap approach to generate the same kind of energy barriers as ours. Indeed if the problem is "tight," meaning that there is just enough room to put all the pieces in the frame, it will be necessary, in order to move a task from one location to another distant location in the frame, to put it in an already occupied location, creating an overlap of the order of the area of the piece to be moved. In the partial solution approach, one has to remove two pieces from the frame. This means that the energy barrier will be from one to two times the energy barrier of the overlap

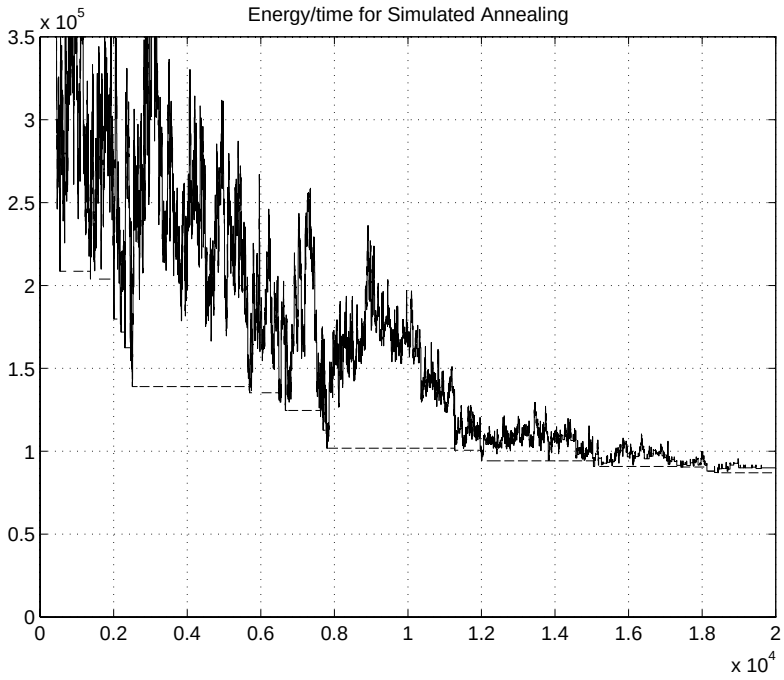

FIG. 7.19.

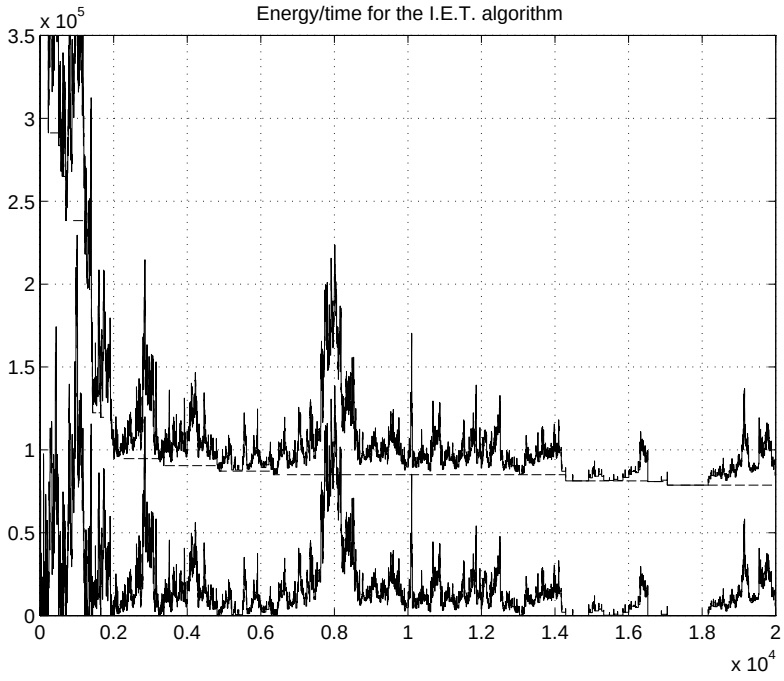

FIG. 7.20.

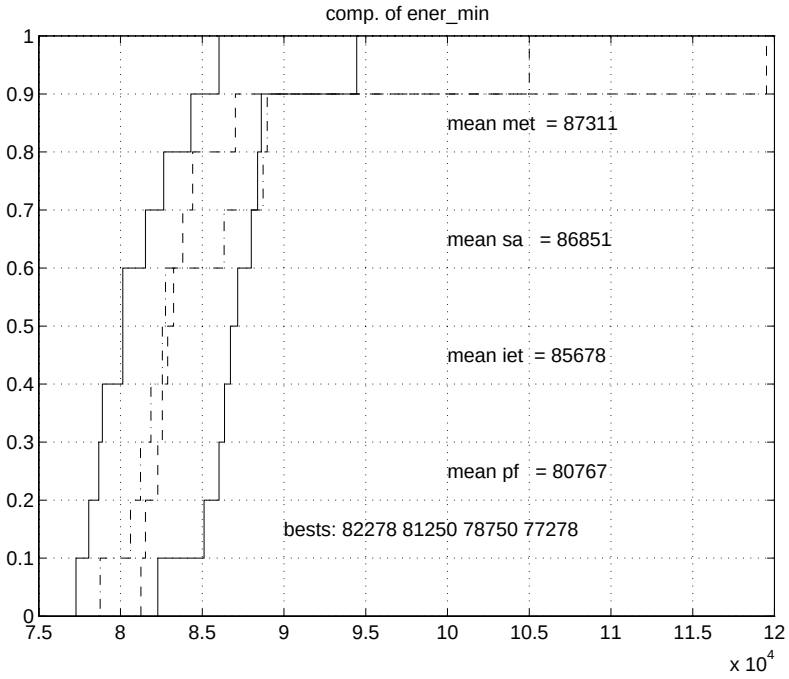

FIG. 7.21.

approach, depending on whether or not the pieces to be moved around are of equal bulk. If one has to exchange a large piece with a number of small pieces, the energy barrier will be the area of the large piece plus the largest area of the small pieces, because once the large piece is removed, the small ones can be transferred one at a time. This is what we can say about the "H" term in the difficulty. Now we have to compare it with the "U" term. For the U term, our approach is better since the energy is quantized: a local minimum solution has at least one piece out; therefore the "U" term is larger than the area of the smallest piece. On the contrary, if space is not discretized, or is finely discretized, the overlap in an imperfect solution of the overlap approach may be arbitrarily small, leading to an arbitrarily large difficulty. Thus the overlap energy cannot safely be used alone. Usually, what is done is to allow pieces to overlap not only between themselves, but also with the outside of the frame—this last kind of overlap being given a specific weight (increasing with time). This is very much like allowing partial solutions, leads to a more complicated algorithm where the balance between two energy terms has to evolve with time, and does not solve completely the problem posed by almost perfect solutions with small scattered overlaps which cannot be improved by local moves.

Let us discuss now the case where some other cost function has to be optimized on the state space of complete solutions. In the partial solution approach, it is usually quite easy to build a cost function which decreases with the number of scheduled tasks and therefore can be used alone. It will be the case, for instance, when the cost function is a sum of negative terms depending on (small) clusters of scheduled tasks. When a task is removed, some of the scheduled clusters are suppressed, the others being unchanged, and so the cost function is increased. It is not so easy and natural to build cost functions which are decreasing with the area of the overlap. Therefore

in the overlap approach an "artificial" overlap term with a big enough weight has to be added to the cost function in the definition of the energy function, creating new energy barriers and increasing the difficulty of the energy landscape.

For all these (qualitative) reasons, we think that the partial solution approach we propose here has some merits, at least in the case of "tight" problems, when it is compared with the traditional overlap approach. This would of course require confirmation by quantitative experimental comparisons.

Conclusion. We touched in this paper on three related topics with various degrees of generality. Our first aim was to bring experimental evidences comforting theoretical results about the behavior of algorithms. We wanted to show that the theory was not concerned with a "never-reached" asymptotic and led to the same qualitative ranking of performances to which an experimental benchmark would lead. Our second aim was to describe a general purpose methodology to deal with scheduling tasks. We insisted on two problems that are likely to be encountered in many situations: the creation of small gaps in the allocation of resources and the way to handle "proportionality constraints."

The third aspect of the paper was to account for experiments on a benchmark of the "jigsaw puzzle" type. Here we were confronted with the practical problem of the choice of parameters and of optimization design options (such as relaxing some of the constraints). Our conclusion on this third point is that we have acquired some know-how about the choice of parameters, which we tried to reflect in section 6, but that we have presently no systematic rule to choose them. We worked very much in a trial and error way, looking at the repartition functions we mentioned, to guide our intuition. A trial and error procedure is somehow justified by the theoretical result that many trials of moderate length are preferable to a long one. This gives us the opportunity to tune the parameters trial after trial.

Anyhow, we have to admit that the choice of parameters requires some skill, especially for the simulated annealing and IET algorithms, where there are more than one parameter to tune. What we did not find too hard to do was, starting from a given Metropolis algorithm at inverse temperature β_{Met}, to find $\beta_{\min} < \beta_{\mathrm{Met}} < \beta_{\max}$ for which simulated annealing performs better than Metropolis. Then we could get some more improvement using the IET algorithm, where again we chose the parameters in relation with those used for simulated annealing. We are not sure at all that this is the best way to tune simulated annealing or the IET algorithm, but it shows at least that the theoretical gains of one algorithm upon the previous one could be obtained in practice. Finally, in the partial freezing method the choice of parameters is easier because the algorithm runs, most of the time, on a restricted state space for which the tuning of parameters is less crucial.

Another positive result of these experiments is that it is possible to get good, if not optimal, solutions even in the case where very nonmonotonous evolutions of the energy are needed, as it is the case here, since the only way to move a piece of the puzzle is to remove it and put it somewhere else afterwards, a succession of two moves the first of which implies an energy increase.

Of course we have touched in this paper on only a limited number of questions. For instance, we leave open the practical question of the best choice of parameters for simulated annealing and for the IET algorithm, since we used only a robust "all purpose" set of parameters, namely, exponential temperature sequences in the case of simulated annealing and logarithmic energy transforms for the IET algorithm. Another question we left purposely in the dark is the choice of elementary moves.

Although it is clear that a benefit can be obtained from the use of more complex compound moves, we felt such an investigation would have been too dependent on the precise examples we chose to study. Rather we tried to lay the stress on general ideas and tools, with the hope that they could be useful in a variety of situations.

Acknowledgments. I wish to thank Professor Robert Azencott for many helpful discussions about this paper. I wish also to thank him for having involved me in the development of an algorithm for an industrial scheduling problem a few years ago. I am also pleased to acknowledge the useful comments of the referees which helped me to improve a first draft of this paper.

REFERENCES

[1] D. Abramson (1992), *A very high speed architecture for simulated annealing,* IEEE Comput., pp. 27–36.

[2] R. Azencott (1988), *Simulated annealing*, Séminaire Bourbaki 40ième année, 1987–1988, p. 697.

[3] R. Azencott (1992), *Sequential simulated annealing: Speed of convergence and acceleration techniques*, in Simulated Annealing: Parallelization Techniques, Wiley Interscience Ser. Discrete Math., R. Azencott, ed., Wiley Interscience, New York, pp. 1–10.

[4] R. Azencott and C. Graffigne (1992), *Parallel annealing by periodically interacting multiple searches: Acceleration rates*, in Simulated Annealing: Parallelization Techniques, Wiley Interscience Ser. Discrete Math., R. Azencott, ed., Wiley Interscience, New York, pp. 81–90.

[5] O. Catoni and R. Cerf (1997), *The Exit path of a Markov chain with rare transitions*, ESAIM Probab. Statist., 1, pp. 95–144; also available online from http://www.emath.fr/Maths/Ps/ps.html.

[6] O. Catoni (1992), *Rough large deviation estimates for simulated annealing: Application to exponential schedules*, Ann. Probab., 20, pp. 1109–1146.

[7] O. Catoni (1998), *The energy transformation method for the Metropolis algorithm Compared with simulated annealing*, Probab. Theory Related Fields, 110, pp. 69–89.

[8] O. Catoni (1991), *Exponential triangular cooling schedules for simulated annealing algorithms: A case study*, in Applied Stochastic Analysis, Proc. US–French Workshop, Rutgers University, New Brunswick, NJ, April 29–May 2, 1991, Lecture Notes in Control and Inform. Sci. 177, I. Karatzas and D. Ocone, eds., Springer–Verlag, Berlin, 1992, pp. 74–89.

[9] C. Cot and O. Catoni (1996), *Piecewise Constant Triangular Cooling Schedules for Generalized Simulated Annealing Algorithms*, preprint, LMENS 96-19; Ann. Appl. Probab., to appear; also available online from http://www.dmi.ens.fr/dmi/preprints.

[10] J. D. Deuschel and C. Mazza (1994), L^2 *convergence of time nonhomogeneous Markov processes:* I. *Spectral Estimates*, Ann. Appl. Probab., 4, pp. 1012–1056.

[11] P. Diaconis and D. Stroock (1991), *Geometric Bounds for Eigenvalues of Markov Chains*, Ann. Appl. Probab., 1, pp. 36–61.

[12] M. Duflo (1996), *Algorithmes Stochastiques*, Mathématiques & Applications (Paris), Springer-Verlag, New York.

[13] M. I. Freidlin and A. D. Wentzell (1984), *Random Perturbations of Dynamical Systems*, Springer-Verlag, New York.

[14] M. R. Garey and D. S. Johnson (1979), *Computers and Intractability: A guide to the theory of NP-completeness*, W. H. Freeman, New York.

[15] S. Geman and D. Geman (1984), *Stochastic relaxation, Gibbs distribution, and the Bayesian restoration of images*, IEEE Trans. Pattern Anal. Mach. Intelligence, 6, pp. 721–741.

[16] C. Graffigne (1992), *Parallel annealing by periodically interacting multiple searches: An experimental study*, in Simulated Annealing: Parallelization Techniques, Wiley Interscience Ser. Discrete Math., R. Azencott, ed., Wiley Interscience, New York, pp. 47–79.

[17] R. Holley and D. Stroock (1988), *Annealing via Sobolev inequalities*, Comm. Math. Phys., 115, pp. 553–559.

[18] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi (1983), *Optimization by simulated annealing*, Science, 220, pp. 621–680.

[19] L. Miclo (1991), *Evolution de l'énergie libre. Application à l'étude de la convergence des algorithmes du recuit simulé*, Doctoral Dissertation, Université d'Orsay, February 1991.
[20] L. Miclo (1996), *Sur les problèmes de sortie discrets inhomogènes*, Ann. Appl. Probab., 6, pp. 1112–1156.
[21] L. Miclo (1995), *Sur les temps d'occupations des processus de Markov finis inhomogènes à basse température*, Stochastics Stochastics Rep., submitted.
[22] A. Trouvé (1993), *Parallélisation massive du recuit simulé*, Doctoral Dissertation, Université Paris 11, January 5, 1993.
[23] A. Trouvé (1994), *Cycle decompositions and simulated annealing*, SIAM J. Control Optim., 34, 1996, pp. 966–986.
[24] A. Trouvé (1995), *Rough large deviation estimates for the optimal convergence speed exponent of generalized simulated annealing algorithms*, Ann. Inst. H. Poincaré Probab. Statist., 32, 1996, pp. 299–348.