7

PARALLEL ANNEALING BY MULTIPLE TRIALS: AN EXPERIMENTAL STUDY ON A TRANSPUTER NETWORK

P. Roussel-Ragot and Gérard Dreyfus

7.1 SUMMARY OF PREVIOUS APPROACHES TO PARALLEL SIMULATED ANNEALING

In an optimization problem one tries to find the best possible state, or configuration, of a system according to a given cost function, which is often referred to as the *energy*. The simulated annealing algorithm is a variant of an iterative improvement method: Starting with an initial state, the algorithm generates a sequence of attempted perturbations, usually termed *elementary moves*. The moves improving the cost function to be minimized are accepted (as they are in classical methods), and the moves increasing the energy are accepted with a probability that depends on the value of a control parameter, the temperature. The value of the temperature is decreased stepwise; the length of each temperature step is controlled by rules that will be

Simulated Annealing: Parallelization Techniques, Edited by Robert Azencott
ISBN 0-471-53231-2

described below. The implementation of simulated annealing on a multiprocessor is not straightforward because of the sequential nature of the method. Simulated annealing is commonly described as a sequence of homogeneous Markov chains: Each computation step of a chain starts only when the previous step is completed. This is the condition for the whole process to lead to a unique feasible configuration.

If an optimal solution is desired, one can perform several annealings independently on different processors. If the initial configurations and/or the sequences of perturbations are independent, it is possible to choose the best configuration among the final configurations. This condition leads to a better configuration than the sequential algorithm with the same computation duration. But, if the computation duration must be significantly shorter, using the parallel algorithm, than with the sequential algorithm this solution is not valid. Two kinds of parallelism can then be used:

1. A parallelism in the evaluation of each move. The computation of a given step of the Markov chain depends only on the configuration of the system before the step and is performed without further interaction with the other steps. Therefore each move evaluation can be parallelized inasmuch as the computations of the variation of the cost function and of the acceptance criterion can be made parallel. This kind of parallelism, which is strongly problem dependent, will not be discussed here.
2. A global parallelism on the Markov chain level. Global parallelism can obviously be combined with the first kind of parallelism, if necessary.

Several attempts to parallelize the simulated annealing algorithm were reported in the literature. The differences lie in the way in which the problem is implemented in parallel, and the main issues are (1) the convergence conditions of the parallel algorithm and (2) the dependence of the parallelism on the problem to be solved.

One class of these parallel methods relies on the geometric properties of the problem to be optimized: In a placement problem, for example, the blocks are distributed among the processors, so that neighbors are gathered together on the same processor at the end of the optimization. This approach was suggested, in particular, by Casotto, Romeo, and Sangiovanni-Vincentelli (1987), Casotto and Sangiovanni-Vincentelli (1987), and Mallela and Grover (1988) in

order to reduce the number of cells to be placed in each subproblem. The placement of the cells in each cluster involves a reduced search space, thus a reduced computation time, with each cluster being evaluated in parallel if desired.

The approach proposed by Darema, Kirkpatrick, and Norton (1987) can be included in this first class because it implies the same kind of deviation from the calculation of the energy: Each processor evaluates one perturbation of the Markov chain under the condition that two processors are not allowed to move the same cells simultaneously. Therefore there is no conflict between processors, and the final configuration is always a valid one. Whenever a perturbation is accepted, the configuration of the cells is updated, regardless of the moves being computed. This method introduces a chaos in the calculation of the energy, since each processor has only the knowledge of the configuration of the system before the parallel perturbations are performed and these perturbations may interact. At low temperature, when the acceptance ratio is low, this does not introduce a large bias compared to the sequential method. At higher temperature the behavior of the parallel method deviates significantly from that of the sequential method.

In the second class of parallel implementations of the simulated annealing algorithm, several perturbations are evaluated in parallel, but only one among the accepted moves is taken into account in the new configuration of the system. This approach was used by Kravitz and Rutenbar (1986), Rutenbar and Kravitz (1986), and Aarts et al. (1986), but the behavior of these methods deviates significantly from the sequential one, since all the rejected moves are counted as steps toward the equilibrium at a given temperature. Aarts et al. (1986) used at high temperature different processors to work on different short Markov chains. In this case when the temperature is changed, one of the final configurations is used as the initial configuration of the next temperature step. As the temperature decreases, fewer Markov subchains are evaluated in parallel, while more processors are used for the generation of each subchain. At low temperature all the processors evaluate a single Markov chain. It is clear that the behavior of this method at high temperature is very different from the behavior of the sequential algorithm.

We propose a method that is different from the approaches of the second class because the steps in the Markov chain are counted in such a way that the number of steps at a given temperature is actually the same as in the sequential mode.

These different methods were implemented on different machines:

1. Darema, Kirkpatrick, and Norton (1987) performed the measurements in a parallel emulation environment, allowing simulation of a shared memory multiprocessor system with up to 64 processors.
2. Aarts et al. (1986) used a parallel machine consisting of 15 Motorola 68000 processors with local memory and a shared 8Mbyte memory.
3. We used a transputer network, with a master processor communicating with all the working processors.

The transputers allow easy hardware and software implementations of parallel architectures. Each transputer can communicate with four neighbors through four high-speed serial links; one link is used for communications between two transputers only, so that the implementation of any parallel architecture does not require any bus protocol. Instructions for communications are provided in the language and no data can be lost since these communications are synchronized. Timesharing is implemented on each transputer, so that waiting time for communications may be used for other calculations.

We have implemented on a transputer network several simulated annealing algorithms: a standard sequential algorithm that gives us the basis for comparisons, a parallel algorithm that exhibits the same convergence properties as the sequential algorithm, and a chaotic parallel algorithm as proposed by Darema, Kirkpatrick, and Norton (1987).

7.2 A PROBLEM-INDEPENDENT PARALLEL IMPLEMENTATION OF SIMULATED ANNEALING

In this section we describe the principle of the parallel implementation of simulated annealing. We first explain and justify the principle of the parallelization. We subsequently propose a statistical model of this stochastic parallel computation.

7.2.1 General Considerations

We first define the acceptance rate $\chi(T)$ as the ratio of the number of accepted moves to the number of attempted moves, averaged over a given temperature. One of the salient features of simulated annealing is the decrease of $\chi(T)$ with temperature. This is due to two facts:

First, the system approaches a minimum, and it is unlikely that a move decreases the energy. Secondly, the value of $\exp(-\Delta E/T)$ becomes very small, so the probability of accepting a move that increases the energy vanishes. Most parallel simulated annealing methods take advantage of this fact: In the low temperature regime, when the acceptance rate χ is small, one can use a number of processor such that $K < 1/\chi$. Thus at most one move will be accepted while K moves are evaluated. As a result the computation time in the parallel mode can be expected to be smaller than the computation time in the sequential mode by a factor of the order of K.

However, the computation time is not the only performance criterion. One wants of course to obtain a valid solution, and perhaps the optimal one (or one of the optimal ones). Therefore the convergence of the algorithm is of central importance. Several theoretical studies of the sequential algorithm have been published (Aarts and van Laarhoven 1985; Geman and Geman 1984), but the theory is much less developed for parallel simulated annealing. However, recent studies (Trouvé, private communication) show that some parallelization schemes may hamper greatly the convergence of the algorithm and even make it impossible in some cases. Therefore it is desirable to design a parallel scheme that is guaranteed to comply with the convergence conditions of the sequential algorithm so that one can capitalize on the accumulated knowledge related to sequential simulated algorithm.

In the following discussion, we suggest a parallel simulated annealing scheme that (1) is problem independent and (2) has the same convergence properties as the sequential algorithm.

7.2.2 Principle

As stated earlier, we use K processors in parallel, each of them evaluating one move. Each processor has its own memory. We want to design a parallel scheme that is equivalent to the sequential one as far as convergence is concerned. The fundamental parameter is the acceptance (or the rejection) rate since it defines the quality of the quasi-equilibrium configuration. We want the parallel scheme to achieve the same acceptance rate as required by the sequential scheme, thus guaranteeing the same convergence behavior.

Two regimes will be considered:

1. A low temperature regime. If $\chi(T) < 1/K$, less than one move out of K will be accepted. Thus in this scheme each processor

attempts moves asynchronously, in parallel, until one of them accepts a move; when an accepted move is found, the processors are synchronized, their memories are updated with the new configuration, and the next evaluation step takes place.

2. A high temperature regime. If $\chi(T) > 1/K$, each processor is allowed to evaluate one move only and waits until all the other processors complete their evaluation. Then one of the accepted moves is chosen at random, the processor memories are updated with the new configuration, and the next evaluation step takes place. We choose one of the accepted moves at random, instead of choosing the first accepted move, because the computation time of a single move can vary substantially; choosing the first move would greatly favor the short computations (e.g., moves of weakly connected blocks, or downhill moves that do not require the computation of the exponential).

In addition to the above mentioned K "slave" processors, the scheme requires one "master" processor that monitors the annealing schedule, chooses the accepted move in the high temperature regime, updates the memory of each processor, and keeps track of statistics.

7.2.3 Models

We use the standard annealing schedule whereby the temperature is decreased stepwise according to $T_{n+1} = \alpha T_n$, where T_n is the nth temperature and α may range from 0.9 to 0.99. We denote by

L_a, the maximum number of accepted moves at a given temperature,

L_t, the maximum number of attempts at a given temperature,

τ_0, the average computation time necessary to evaluate one move in the sequential mode.

The temperature is decreased either when the number of accepted moves at the current temperature reaches L_a or when the number of attempted moves at the current temperature reaches L_t, whichever limit is reached first. Note that τ_0, the average computation time to evaluate one move in the sequential mode, is not exactly the same as that in parallel because the master processor can take care of the necessary statistics while the slave processors evaluate the moves.

Therefore τ_0 is an upper limit of the average computation time for one move in the parallel implementation.

Model of the High Temperature Mode In the high temperature mode each processor evaluates one move, and all processors are synchronized at the end of each evaluation. We denote by τ_r the average overhead due to communications with the K slaves and their synchronization. Therefore the average time necessary for the K processors to perform one evaluation is $\tau_0 + \tau_r$.

Since the length of the Markov chain depends on the number of accepted moves and/or on the number of attempted moves, we first have to evaluate these quantities. Assume that, after one parallel evaluation of K moves, r moves out of K are rejected; $K - r$ moves are found acceptable, but only one of them will be actually accepted in the Markov chain, the other ones being discarded. In the serial mode the ratio of the number of accepted moves to the number of attempted moves would be $(K - r)/K$; however, in the parallel mode only one move is accepted. Therefore, if we want to preserve the convergence behavior of the serial algorithm (Catoni and Trouvé, 1989), we must consider that the effective number of attempted moves n^* is such that

$$n^* = \frac{K + 1}{K - r + 1}.$$

When no move is accepted, then $n^* = K$. This is a good estimate of the effective number of attempted moves since, on the average, the ratio of the number of accepted moves to the effective number of attempted moves is equal to the sequential acceptance rate:

$$\frac{\sum_{i=1}^{K}\binom{K}{i}\chi^i(1-\chi)^{K-i}}{(1-\chi)^K K + \sum_{i=1}^{K}\binom{K}{i}\chi^i(1-\chi)^{K-i}(K+1)/(i+1)} = \chi.$$

We now evaluate the total effective number of attempted moves N^* and the total number of accepted moves taken into account, N_a^*, once N parallel evaluations of K moves have been performed. The probability that r moves out of K are rejected is equal to

$$\binom{K}{r}(1-\chi)^r\chi^{K-r};$$

the probability that all K moves are rejected is $(1-\chi)^K$. Therefore the average effective number of attempts is given by

$$N^* = N\left[(1-\chi)^K K + \sum_{i=1}^{K} \binom{K}{i} \chi^i (1-\chi)^{K-i} \frac{K+1}{i+1}\right],$$

or equivalently

$$N^* = N\frac{1-(1-\chi)^K}{\chi}.$$

The limit of L_t attempted moves is reached after a number N_t of parallel evaluations of K moves, which is given by

$$N_t = L_t \frac{\chi}{1-(1-\chi)^K}.$$

The number N_a^* of accepted moves taken into account is equal to the number of parallel evaluations of K moves leading to at least one acceptable move; hence

$$N_a^* = N\left[1-(1-\chi)^K\right].$$

The limit of L_a accepted moves is reached after a number N_a of parallel evaluations of K moves, which is given by

$$N_a = \frac{L_a}{1-(1-\chi)^K}.$$

Therefore in the high temperature mode the number of parallel evaluations at a given temperature is

$$N_p = \min(N_t, N_a).$$

The corresponding computation time is

$$t_p = N_p(\tau_0 + \tau_r).$$

In the serial mode the computation time is

$$t_s = \tau_0 \frac{L_a}{\chi} \quad \text{if the } L_a \text{ limit is reached first,}$$

$$t_s = \tau_0 L_t \quad \text{if the } L_t \text{ limit is reached first.}$$

Therefore, whichever limit L_a or L_t is reached first,

$$\frac{t_p}{t_s} = \frac{\chi}{1 - (1 - \chi)^K} \left(1 + \frac{\tau_r}{\tau_0}\right). \tag{7.1}$$

Note that $\lim(t_p/t_s) = 1 + \tau_r/\tau_0$ when $\chi \to 1$ and that $\lim(t_p/t_s) = (1 + \tau_r/\tau_0)/K$ when $\chi \to 0$.

At high temperature the efficiency is low; at low temperature the computation time is roughly divided by the number of processors, as expected, if the overhead time τ_r is small compared to τ_0. The average value of τ_0 is known from the serial implementation of the simulated annealing algorithm. The determination of τ_r is not straightforward and depends on the problem. If τ_0 is constant, τ_r may be approximated by K times the communication time: If the K parallel computations end at the same time, K successive communications will be required for the master to know all the results and restart the slaves.

Model of the Low Temperature Mode In the low temperature mode each processor evaluates moves independently until one move is accepted. At the end of each individual evaluation, the processors send their results to the master. If no move was accepted by any processor since the previous communication, another move is attempted. If one move was accepted, the memories of the slave processors are updated, and the processors are synchronized. In this mode all the rejected moves are counted as steps toward equilibrium.

To model the behavior of this low temperature mode, it is necessary to evaluate the number of moves required for one move to be accepted. This can be done in two ways. Aarts et al. (1986) evaluate the number of parallel calculations required. Their estimation leads to a number of parallel calculations equal, on the average, to $1/(1 - (1 - \chi)^K)$. Since it is easier to estimate the time characteristics of individual moves from the sequential results, we find it preferable to evaluate the number of such moves. One configuration is accepted on the average when $1/\chi$ moves are evaluated. The process

has then performed $1/\chi$ steps toward equilibrium in the serial mode. Since we want to obtain a feasible configuration of the system, if another move is accepted by one of the $K-1$ other processors, we do not take it into account. When the processors are synchronized, $1/\chi + K - 1$ moves have been evaluated, but we count only $1/\chi + (K-1)(1-\chi)$ steps in the Markov chain since we discard all the accepted moves but one.

If τ_m is the time required to obtain one accepted move in parallel, we can model the behavior of the low temperature mode as follows:

1. When L_a is reached first, $t_p = L_a\tau_m$ and $t_s = L_a\tau_0/\chi$, thus

$$\frac{t_p}{t_s} = \chi\frac{\tau_m}{\tau_0} \tag{7.2}$$

2. When L_t is reached first,

$$t_p = \frac{L_t}{1/\chi + (K-1)(1-\chi)}\tau_m \tag{7.3}$$

and $t_s = L_t\tau_0$, thus

$$\frac{t_p}{t_s} = \frac{1}{1/\chi + (K-1)(1-\chi)}\frac{\tau_m}{\tau_0}.$$

Note that for $K = 1$, one has $\tau_m = \tau_0/\chi$ so that $t_p = t_s$, as expected.

Here again, the average value of τ_0 is known, but the determination of τ_m is not an easy task and depends on the problem. If τ_0 is constant and if $1/\chi$ is much larger than K, the value of τ_m can be approximated by $(\tau_0 + \tau_c)/K\chi$, τ_c being the time required by one slave to communicate its result to the master. This approximation would not be valid for small values of $1/\chi$. If the first accepted move is, on the average, the second attempted move, it would mean that the first and the third moves may be accepted with nonzero probability. If the first move is accepted, τ_{m1} is equal to $\tau_0 + K\tau_c$; if the second or the third move is accepted, $\tau_{m2} = 2\tau_0 + 2\tau_c$, assuming that $K\tau_c < \tau_0$, since the synchronization occurs when all the active processors have achieved their computations. Thus the estimated value of τ_m would be erroneous because $\tau_{m2} - \tau_{m1} = \tau_0 + (2-K)\tau_c$ may be large com-

pared to τ_{m2}. If $1/\chi$ is much larger than K, $\tau_0 + (2 - K)\tau_c$ is small compared to $\tau_0/K\chi$. Moreover, since the low temperature regime will be used only if $1/\chi \gg K$ (i.e., at low temperature or for a small number of processors), this approximation is valid.

7.3 RESULTS

7.3.1 A Simple Placement Problem

We tested our parallel methods and models on a simple placement problem. It consists of a two-dimensional array of b^2 chips arranged on a square grid. In the ground state configuration of the system, each chip is connected to its nearest neighbours by two-terminal connections. The elementary move is the exchange of two chips chosen at random; the cost function is equal to the total length of the wires. Its minimum value is equal to $2b(b - 1)$. The parameters of the standard sequential annealing schedule are the following:

1. The initial configuration is chosen at random.
2. The initial temperature is chosen so that the acceptance rate is larger than 0.9.
3. The temperature is modified when $5b^2(b^2 - 1)$ moves have been evaluated or when $b^2(b^2 - 1)/2$ moves have been accepted.
4. The cooling parameter α is equal to 0.9.
5. The simulated annealing process is stopped when the temperature reaches the value 0.2 or when no move is accepted at a given temperature.

This annealing schedule was not intended to be optimal. We only wanted to evaluate the performances of the parallel algorithm as compared to those of the serial algorithm, subject to the same conditions. Experiments were performed with 25, 49, and 81 chips on three and six transputers. We present the most relevant results here, obtained on 81 chips in two cases: the communication time is small compared to the computation time, and the communication and computation times are of the same order of magnitude.

7.3.2 Numerical Results

Both temperature modes were investigated independently on a complete annealing, although they are not intended to be actually used on the whole temperature range. We compared the behavior of the high

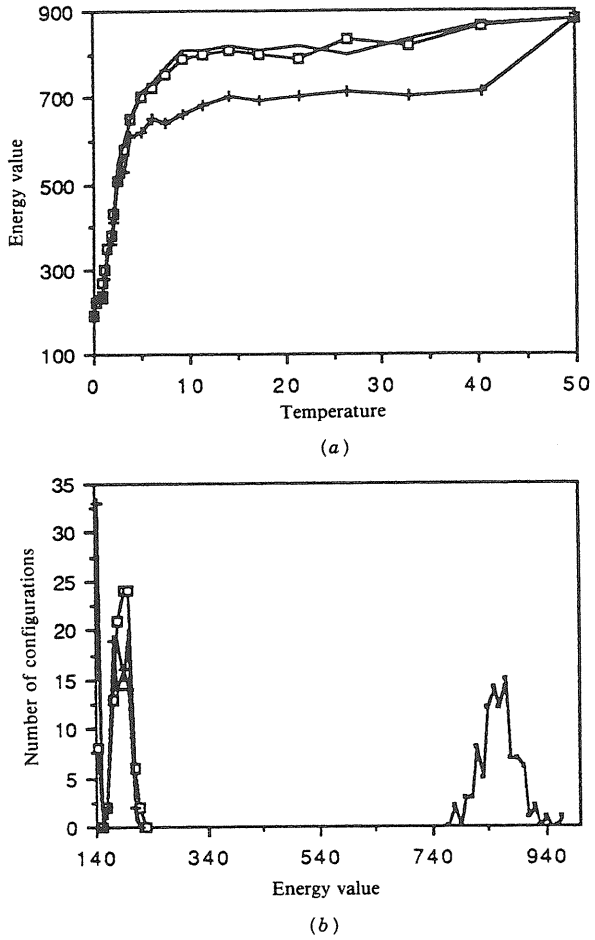


Figure 7.1 (*a*) Energy at the end of a temperature step for the sequential mode (squares) and the high temperature (dots) and low temperature (crosses) modes. (*b*) Energy distribution of the initial configurations (right-hand peak), and energy distributions of the configurations after annealing (left-hand peaks).

and low temperature modes to the behavior of the sequential mode, on 100 different initial configurations. Figure 7.1*a* is a plot of the average final energy of each temperature step as a function of temperature. In the figure it can be seen that the high temperature mode behaves similarly to the sequential mode, whereas the low temperature mode decreases the energy quickly for high temperature values. This is due to the fact that in the low temperature mode, the first accepted move is taken into account for updating the system. Since we used T414 transputers without floating-point computations, the computation of the exponential is long as compared to the execution of a

simple instruction. Thus the first accepted move often happens to be a move that decreases the energy. At low temperature the annealing curve is the same for the three modes. This allows us to switch from high to low temperature mode when the low temperature mode becomes more efficient, still complying with the quality of convergence of the sequential algorithm. The final and initial energy distributions

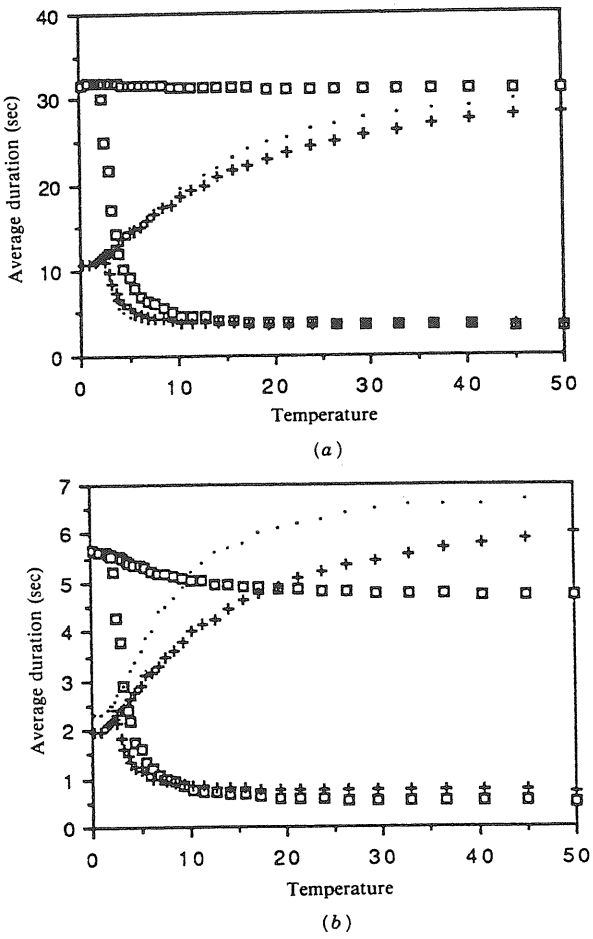
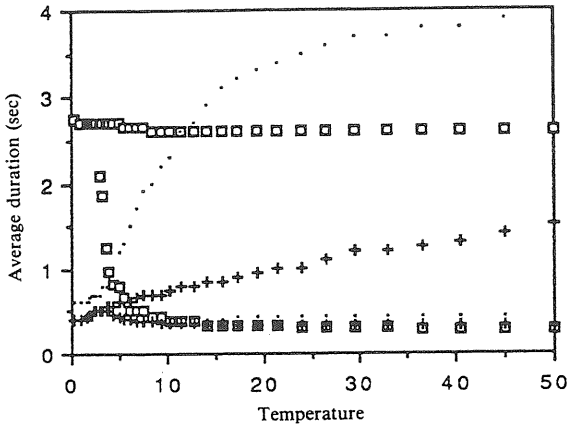


Figure 7.2 Average duration of a temperature step. Results were obtained in the sequential mode (squares), high temperature mode (dots), and low temperature mode (crosses): Temperature decreased when L_t moves have been attempted (upper three curves); temperature decreased when L_a moves have been accepted (lower three curves). (*a*) Overhead time much smaller than computation time; measurements performed with three T414 transputers. (*b*) Overhead time of the same order of magnitude as computation time; measurements performed with three T414 transputers. (*c*) Overhead time of the same order of magnitude as computation time; measurements performed with six T800 transputers.



(c)

Figure 7.2 *(Continued)*

are shown in Figure 7.1*b*. No significant difference between the three modes can be observed because of the nature of the problem we investigated; a difference should appear when using finely tuned annealing schedules, since the low temperature mode exhibits significant deviation at high temperature.

The computation times, averaged on 10 experiments, for each temperature step of the annealing process for the sequential and the parallel algorithms, are shown in Figure 7.2. On all diagrams the upper three curves are the average duration of a temperature step measured if the temperature is decreased when L_t moves have been attempted. The lower three curves are the average duration of a temperature step if the temperature is decreased when L_a moves have been accepted. As expected, the duration of a temperature step in the sequential mode is virtually constant in the first case and increases sharply at low temperature in the second case. We find that the acceleration is poor at high temperature. The time required for the parallel algorithms is even higher than the time required for the sequential algorithm when the communication and synchronization time is close to the computation time (Figure 7.2*b* and *c*). At low temperature the duration is almost divided by the number of processors. It can be seen in Figure 7.2*c* that the average duration of a temperature step, when the limit L_t is used, is small at high temperature for the low temperature mode. This is due to the fact that in this mode, as mentioned before, the energy decreases quickly at high temperature. Thus the acceptance rate decreases as well, and the

number of rejected moves is high. Moreover all the rejected moves are counted as steps toward equilibrium, and the limit L_t is then reached sooner. When the L_a limit is used, the overall annealing time is divided by 2 with three processors when the communication time is large, and by 2.5 when it is small. This value depends strongly on the annealing schedule: If more time is spent at low temperature, as is

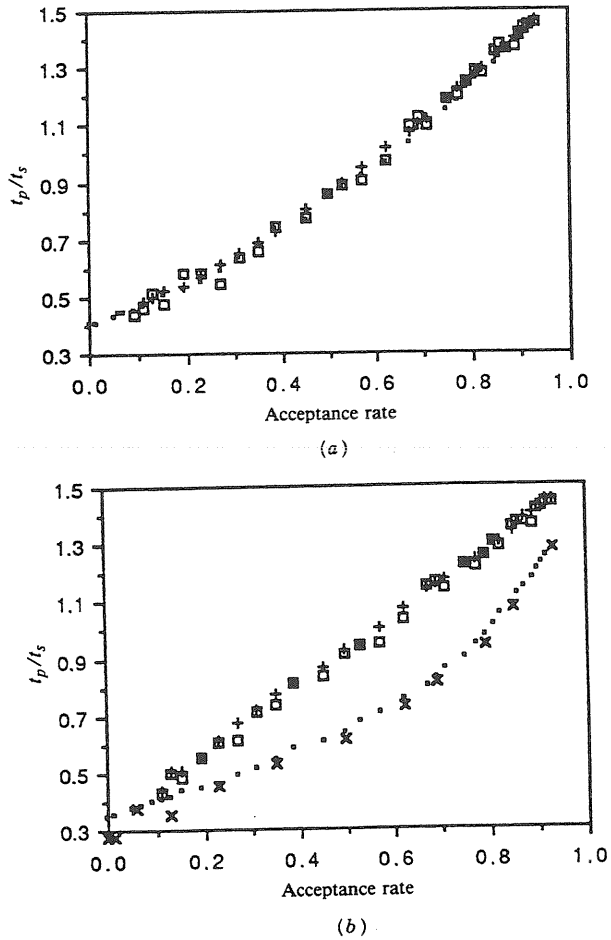


Figure 7.3 Ratio of the duration of a temperature step in the parallel mode to the duration of a temperature step in the sequential mode. Measurements were performed with three transputers, with overhead of the same order of magnitude as computation time: measurements results (squares), results computed from the model when the L_a limit is used (crosses), measurements results (black squares), results computed from the model when the L_t limit is used (crisscrosses). (*a*) High temperature mode. (*b*) Low temperature mode.

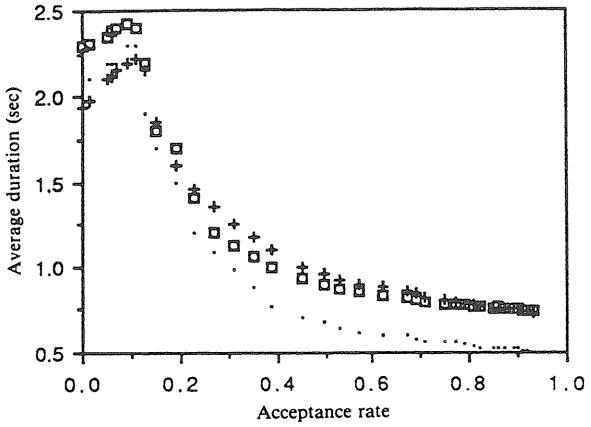


Figure 7.4 Average duration of a temperature step. Results were obtained in the high (squares) and low (crosses) temperature modes and the chaotic parallel mode (dots) proposed by Darema, Kirkpatrick, and Norton (1987); temperature decreased as L_a moves were accepted.

frequently the case in real optimization problems, the overall speedup factor will be higher.

Figure 7.3 exhibits very good agreement between the measurements performed in the parallel temperature mode and the estimated values of t_p/t_s when the L_a limit is reached first (relations 1 and 2), and when the L_t limit is reached first (relations 1 and 3). The acceleration is higher when the L_t limit is used in the low temperature mode. But since each temperature step is much longer than in the case of the L_a limit, it is definitively worthwhile to use the L_a limit when it is reached first. To compute the estimates from the model, we evaluated $\tau_0 + \tau_r$ as the average duration of a temperature step divided by the number of parallel evaluations, and τ_m as the average duration of a temperature step divided by the number of accepted moves. The average value of $\chi(T)$ was estimated from the sequential results.

The computation times, when the L_a limit is reached first, averaged on 10 experiments for each temperature step of the annealing process for the parallel algorithms and for the chaotic parallel algorithm proposed by Darema, Kirkpatrick, and Norton (1987) are shown on Figure 7.4. It appears that the chaotic method is more efficient at high temperature when the L_a limit is used, which is due to the fact that several moves are accepted when one parallel evaluation is performed. At low temperature its efficiency is lower than the efficiency of the low temperature mode because few moves are accepted and time is wasted trying to draw different pairs of blocks. It can be noted that this method does not build a Markov chain because more than

one modification is taken into account for each step. With the simplified placement problem the quality of convergence is the same as with the sequential algorithm, but the theoretical proofs of convergence cannot be used unchanged.

7.4 CONCLUSION

We have proposed a problem-independent parallel implementation of the simulated annealing algorithm that guarantees the same quality of convergence as the sequential algorithm. The parallel algorithm consists of two modes: One is intended to be used at high temperatures (at least one move is accepted when K moves are evaluated) and the other one is to be used at low temperatures (at most one move is accepted). Statistical models of both modes have been derived and compared to experiments on a simple placement problem, implemented on a network of Transputers. The architecture of the system consists of a "master" processor linked to K "slave" processors. These models are expressed as functions of the acceptance rate, which enables an estimation of the acceleration for any optimization problem. They take into account the fact that depending on the implementation of the sequential algorithm, the length of the Markov chain for each temperature step can be taken either equal to a given number of accepted moves or equal to a given number of attempted moves. The condition for switching from the high temperature mode to the low temperature mode depends on the number of processors used and on the length of the Markov chain. Further improvements in this technique might be achieved if the processors are allowed to proceed asynchronously, that is, if they are not synchronized each time a move is accepted. The chaotic parallel method proposed by Darema, Kirkpatrick, and Norton (1987) is compared to the high and low temperature modes. It exhibits better performance mostly at high temperature but introduces a significant difference from the sequential algorithm, which can lead to unpredictable results on different optimization problems.

ACKNOWLEDGMENTS

The authors are very grateful to A. Trouvé and O. Catoni (see Chatper 9 in this volume) for their critical comments on the derivation of the effective number of moves.

REFERENCES

Aarts, E. H. L., and P. J. M. van Laarhoven. Statistical cooling: A general approach to combinatorial optimization problems. *Philips J. Res.* **40** (1985): 193–226.

Aarts, E. H. L., F. M. J. de Bont, J. H. A. Habers, and P. J. M. van Laarhoven. Parallel implementations of the statistical cooling algorithm. *Integration VLSI J.* **4** (1986): 209–238.

Casotto, A., and A. Sangiovanni-Vincentelli. Placement of standard cells using simulated annealing on the connection machine. *Proc. IEEE Int. Conf. Computer Design*, 1987, pp. 350–353.

Casotto, A., F. Romeo, and A. Sangiovanni-Vincentelli. A parallel simulated annealing algorithm for the placement of macro-cells. *IEEE Trans. CAD* **CAD-6**, 5 (1987): 838–847.

Catoni, O., and A. Trouvé. Parallel annealing by multiple trials: a mathematical study. Chapter 9 in this volume.

Darema, F., S. Kirkpatrick, and V. A. Norton. Parallel algorithms for chip placement by simulated annealing. *IBM J. Res. Develop.* **31**, 3 (1987): 391–402.

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

Kirkpatrick, S., C. Gelatt, Jr., and M. Vecchi. Optimization by simulated annealing. *Science* **220** (1983): 671–680.

Kravitz, S., and R. Rutenbar. Multiprocessor-based placement by simulated annealing. *Proc. 23th ACM/IEEE Design Automation Conf.*, 1986.

Lam, J., and J.-M. Delosme. Performance of a new annealing schedule. *Proc. 25th AMC/IEEE Design Automation Conf.*, 1988, pp. 306–311.

Mallela, S., and L. Grover. Clustering based simulated annealing for standard cell placement. *Proc. 25th ACM/IEEE Design Automation Conf.*, 1988, pp. 312–317.

Otten, R., and L. van Ginneken. Stop criteria in simulated annealing. *Proc. IEEE Int. Conf. Computer Design*, 1988, pp. 549–552.

Rutenbar, R., and S. Kravitz. Layout by annealing in a parallel environment. *Proc. IEEE Conf. Computer Design*, 1986, pp. 434–437.