TM-1147

Temperature Parallel Simulated Annealing – Application to Biological Problems

by
M. Hirosawa & M. Ishikawa

Janualy, 1992

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03)3456-3191 ~ 5
Telex ICOT J32964

Institute for New Generation Computer Technology

Temperature Parallel Simulated Annealing —Application to Biological Problems—

MAKOTO HIROSAWA AND MASATO ISHIKAWA
Institute for New Generation Computer Technology
1-4-28 Mita, Minato-ku, Tokyo 108, Japan

Introduction

We propose a parallel simulated annealing algorithm that can be applied to solving biological problems. Simulated annealing (SA) is a stochastic algorithm used to solve complex combinatorial optimization problems [1]. It searches for a global optimal solution in solution space without being captured in local optima.

SA simulates the annealing process of physical systems using a parameter, *temperature* and an evaluation value, *energy*. At high temperatures, the seach point in solution space jumps out of a local energy minimum. At low temperatures, the point falls to the nearest local energy minimum.

An outline of the SA algorithm is as follows. Given an arbitrary initial solution x_0, the algorithm generates a sequence of solutions $\{x_n\}_{n=0,1,2,\ldots}$ iteratively, finally outputting x_n for a large enough value of n. In each iteration, the current solution x_n is randomly modified to get a candidate x'_n for the next solution, and the variation of the energy $\Delta E = E(x'_n) - E(x_n)$ is calculated to evaluate the candidate. When $\Delta E \leq 0$, the modification is good enough to accept the candidate: $x_{n+1} = x'_n$. When $\Delta E > 0$, the candidate is accepted with probability $p = \exp(-\Delta E/T_n)$, but rejected it if $x_{n+1} = x_n$, where $\{T_n\}_{n=0,1,\ldots}$ is a *cooling schedule*, a sequence of temperatures decreasing with n.

Because the solution x_n is distributed according to the Boltzmann distribution at temperature T, the distribution converges to the lowest energy state (optimal solution) as the temperature decreases to zero. Thus, one might expect SA to be capable of providing the optimal solution, in principle. It is well-known that the cooling schedule has great influence on SA performance. Here arises the *cooling schedule problem*.

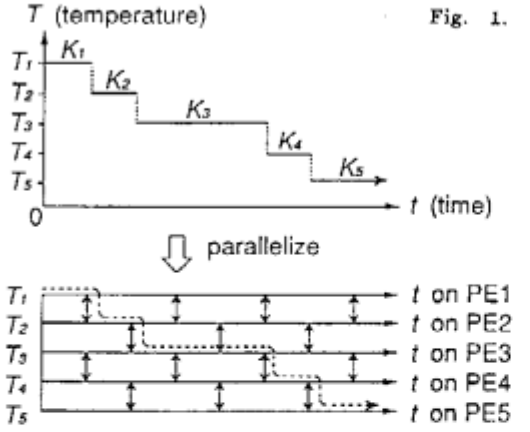

Fig. 1.

Temperature parallel SA

The basic idea behind the algorithm is to use parallelism in temperature, to perform annealing processes concurrently at various temperatures [2]. The algorithm automatically constructs an appropriate cooling schedule from a given set of temperatures (Fig.1). Hence it partly solves the cooling schedule problem.

The outline of the algorithm is as follows. Each processing element (PE) maintains one solution and performs the annealing process concurrently at a *constant* temperature that differs between PEs. After every k annealing steps, each pair of PEs with adjacent temperatures performs a *probabilistic exchange of solutions*. The probability of the exchange between two solutions is defined as follows:

$$p(\Delta T, \Delta E) = \begin{cases} 1 & \text{if } \Delta T \cdot \Delta E < 0 \\ \exp(-\frac{\Delta T \cdot \Delta E}{TT}) & \text{otherwise.} \end{cases}$$

The probability has been defined such that it is advantageous to solutions having low energies. Hence, the solution at the lowest temperature is expected to be the best solution so far. The cooling schedule is invisibly embedded in the parallel execution.

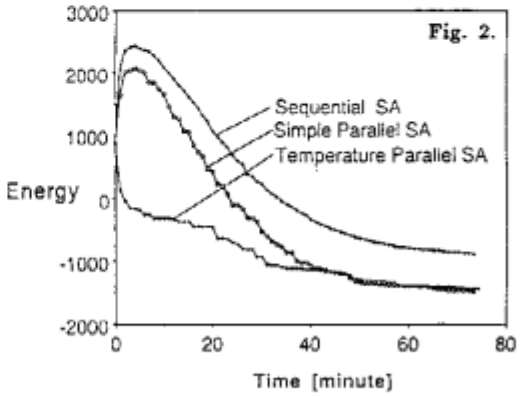

Fig. 2.

Application to biological problems

We have applied the temperature parallel SA to two biological problems: one is protein folding [3] and the other is multiple sequence alignment [4]. The system is implemented on the Multi-PSI [5], an MIMD parallel machine having 64 PEs. We compared the histories of the energy given by the parallel algorithm with those given by sequential and simple parallel SA (Fig.2 [4]).

The two parallel SA algorithms can obtain better energy values during the entire annealing process. It was reported that the temperature parallel algorithm gave better results than the simple parallel algorithm [2]. On a multiple alignment problem, however, no significant difference was found in the annealed results obtained with the two parallel algorithms.

Nevertheless, the temperature parallel algorithm is advantageous. We can stop the execution at any time and examine whether a satisfactory solution has already been obtained. If a solution has not yet to be reached, we can resume the execution *without any rescheduling* to obtain a better solution. In contrast, in the simple parallel algorithm, when an obtained solution is not satisfactory, we have to reconsider the cooling schedule and repeat the time-consuming process.

[1] Kirkpatrick,S. Gelatt,C.D. and Vecci,M.P. (1983) Optimization by simulated annealing. *Science, no.4598.*

[2] Kimura,K. and Taki,K. (1990) Time-homogeneous parallel annealing algorithm. *Proc. Comp. Appl. Math. (IMACS'91)*, 13, 827–828.

[3] Hirosawa,M., Feldmann,R.J., Rawn,D., Ishikawa,M., Micheals,G. and Hoshida,M. (1992) Folding simulation using temperature parallel simulated annealing. *Fifth Generation Computer Systems (FGCS'92)*, (in preparation).

[4] Ishikawa,M., Tomoyuki,T., Hoshida,M., Nitta,K., Ogiwara,A. and Kanehisa,M. (1991) Multiple alignment by parallel simulated annealing. *Genome Informatics Workshop II*, 112–115, (in Japanese).

[5] Nakajima,K., Inamura,Y., Ichiyoshi,N., Rokusawa,K. and Chikayama,T. (1989) Distributed implementation of KL1 on the Multi-PSI/V2. *Proc. 6th Int. Conf. on Logic Programming.*