Synchronous and Asynchronous Parallel Simulated Annealing with Multiple Markov Chains

Soo-Young Lee, *Senior Member, IEEE*, and Kyung Geun Lee, *Member, IEEE*

Abstract—Simulated annealing is a general-purpose optimization technique capable of finding an optimal or near-optimal solution in various applications. However, the long execution time required for a good quality solution has been a major drawback in practice. Extensive studies have been carried out to develop parallel algorithms for simulated annealing. Most of them were not very successful, mainly because multiple processing elements (PEs) were required to follow a single Markov chain and, therefore, only a limited parallelism was exploited. In this paper, we propose new parallel simulated annealing algorithms which allow multiple Markov chains to be traced simultaneously by PEs which may communicate with each other. We have considered both synchronous and asynchronous implementations of the algorithms. Their performance has been analyzed in detail and also verified by extensive experimental results. It has been shown that for graph partitioning the proposed parallel simulated annealing schemes can find a solution of equivalent (or even better) quality up to an order of magnitude faster than the conventional parallel schemes. Among the proposed schemes, the one where PEs exchange information dynamically (not with a fixed period) performs best.

Index Terms—Asynchronous communication, graph partitioning, multiple Markov chains, parallel algorithm, simulated annealing, solution quality, speed-up.

1 Introduction

Simulated annealing (SA) is an iterative probabilistic algorithm which combines local search with Monte Carlo techniques [1]. Numerous researchers have demonstrated that SA can be very effective in many optimization problems such as TSP, placement and routing in VLSI design, logic minimization, code design and image processing, etc. One of the distinct features of the SA is its hill-climbing capability which allows escape from local optima. Also, it may start from any initial solution and converges to an optimal or near optimal solution [2]. In addition, SA is applicable to various classes of problems including discrete, nondifferentiable, or combinatorial problems, etc. However, the long execution time of SA has been the major drawback in practice.

There have been numerous efforts to make SA practical for problems of realistic size by shortening the execution time. These efforts can be classified into two major categories: algorithmic optimization (of sequential SA) and parallel processing.

Two types of approaches are prevalent in the category of optimizing SA: careful perturbation and cooling. In the approach of careful perturbation, the idea of range-limiting was introduced and successfully applied to function minimization and VLSI design [3], [4]. Since the execution time is proportional to the total number of trials, a heuristic perturbation scheme may shorten the execution time by reducing the chance of rejection [3], [5]. In the other approach, most schemes utilize statistical quantities such as mean and variance of costs obtained during annealing. They usually modify the conventional annealing schedule to accelerate the execution of SA. However, it has been shown that modifying the annealing schedule does not improve both the solution quality and execution time [6], [7]. The execution time was reduced by a factor of 2 at best.

The advent of parallel computers has made it possible to achieve speed-up of orders of magnitude in various applications. Most parallel SA schemes previously proposed share the feature that multiple PEs follow a single search path (Markov chain) [8], [9], [10], [11], [12], [13] (to be referred to as SMC PSA). In SMC PSA, a perturbation at each PE and/or evaluation of the perturbation is executed in parallel, and then only one global solution is accepted in each iteration. While this approach preserves the sequential nature of SA, following a single search path might be an unnecessary restriction especially from the viewpoint of performance, i.e., the execution time and solution quality. That is, only a limited parallelism is exploited, which is believed to be the main reason why most of the previous PSAs have not been very successful.

Banerjee et al. implemented SMC PSA schemes for a standard cell placement problem on hypercube multiprocessors. The PSA schemes are based on a sequential version of SA (TimberWolf). The task (SA) is spatially partitioned and non-exact calculations by multiple PEs are allowed by assuming temporary locality. They concluded that such *erroneous* calculations can achieve a considerable speed-up without seriously affecting convergence [10]. However, this may be acceptable

• *S.-Y. Lee is with the Department of Electrical Engineering, Auburn University, Auburn, AL 36849. E-mail: sylee@eng.auburn.edu.*
• *K.G. Lee is with Samsung Electronics Co., Network Research Group, Seoul, Korea 138-160. E-mail: kglee@metro.telecom.samsung.co.kr.*

Manuscript received July 30, 1992; revised Dec. 5, 1994.
For information on obtaining reprints of this article, please send e-mail to: transpds@computer.org, and reference IEEECS Log Number D95138.

for particular applications only when a periodic synchronization is incorporated to compensate for the erroneous calculations. Consequently, either the synchronization overhead may increase the execution time or the erroneous calculations may deteriorate the solution quality. Also, a general purpose PSA scheme which takes advantage of the acceptance rate but follows a single Markov chain was proposed by Roussel-Ragot and Dreyfus. Besides the drawbacks due to the single Markov chain, their synchronization strategy using the acceptance rate does not appear to be very efficient [11].

The feasibility of solving combinatorial problems on a Boltzmann machine was investigated by Aarts and Korst. [15]. However, it has been shown by computer simulation that the performance characteristics (convergence and speed) strongly depend on the type of problem. Also, a practical implementation on a massively parallel machine resulted in very small speed-up for placement problems [16]. These facts imply that the implementation of SA on a massively parallel machine is not yet suitable in practice due to its sequential and random natures.

In contrast to the abundance of SMC PSA schemes, there are very few schemes where PEs are allowed to follow multiple Markov chains (to be referred to as MMC PSA). Aarts and Korst described the idea of the MMC PSA, namely the *division algorithm*. However, it does not seem to have been fully developed. The idle and communication times which are inevitable in practice are not taken into account. In addition, PEs may exchange the information at non-quasi-equilibrium state in the proposed algorithm, which may not lead to an optimal solution [7].

In this paper, we propose a new MMC PSA which can overcome the potential drawbacks of the previous PSA schemes. One of the distinct features of our scheme is that PEs (following multiple Markov chains) communicate with each other dynamically, not with a fixed period. This yields a significant savings in communication time by allowing PEs to exchange only useful information. Also, it is shown that the information exchange can be implemented asynchronously in order to further improve performance of the proposed MMC PSA. The goals of the proposed PSA are

1) to achieve higher speed-up and efficiency,
2) to obtain a good solution quality (at least as good as that of the sequential SA), and
3) to realize problem independent performance.

This paper is organized as follows. In Section 2, the new MMC PSA and its asynchronous version are described, following our versions of SMC PSAs. In Section 3, The SMC and MMC PSAs are compared through theoretical analysis. The implementation results for graph partitioning are provided with discussions in Section 4, followed by conclusions in Section 5.

2 Parallelization of Simulated Annealing

2.1 Simulated Annealing

SA, also called *statistical cooling*, has many attractive features including the convergence to a high quality solution, versatility, and ease of implementation. However, SA requires a careful setup of the basic strategies and parameters, e.g.,

1) how to define the configuration space and an appropriate cost function,
2) how to implement an efficient perturbation scheme and
3) how to design the cooling schedule including *initial temperature*, a rule for temperature decrement, *quasi-equilibrium conditions* for a temperature, and a *stop criterion*.

A typical structure of SA consists of two nested loops as shown in Fig. 1. It starts from an arbitrarily selected configuration s_0 with an appropriate initial temperature (T_i) and works to minimize a given *cost function*. At a fixed temperature, the inner loop repeatedly executes the following three step operation, to be referred to as *iteration*, until an inner loop break condition is satisfied. It randomly perturbs the current solution (or configuration), evaluates the corresponding cost, and accepts the new solution with the probability of $\min\left(1, e^{-\frac{\Delta C}{T}}\right)$ where ΔC is the cost change due to the perturbation and T is the current temperature. The outer loop decreases temperature according to the rule, $T \leftarrow \alpha \cdot T$, where α, the *cooling coefficient*, satisfies $0 < \alpha < 1$. It can be said that SA consists of a sequential chain of consecutive perturbation, evaluation and decision steps.

```
s ⟵ s0 ;
T ⟵ Ti;
While the outer loop break condition is not met do {
    While the inner loop break condition is not met do {
        Perturb current configuration s to s*;
        Evaluate cost function and find the difference;
                ΔC = C(s*) − C(s);
        Accept new configuration with probability min(1, e^(−ΔC/T));
    }
    T ⟵ α T;
}
```

Fig. 1. Typical structure of simulated annealing.

Those parameters mentioned above, which control the execution of the nested loops are called *scheduling parameters*, i.e., *initial temperature* (T_i), *cooling coefficient* (α), and *equilibrium conditions* (also known as *break conditions*) for the inner and outer loops. The execution time and solution quality are heavily dependent on the scheduling parameters. In the following, we will first describe our versions of SMC PSAs, mainly for comparison purposes, and then present the proposed MMC PSA. In evaluating performance of a PSA, we need to consider solution quality as well as execution speed. The execution speed may be quantified in terms of *speed-up* (S) and *efficiency* (E). The speed-up (S) is defined as the ratio of the execution time (on one PE) by the sequential SA to that by the PSA (on N PEs) for an equivalent solution quality. In the ideal case, S would be equal to N. Efficiency (E) is defined as the ratio of the actual speed-up to the ideal speed-up (N).

2.2 Single Markov Chain PSA

Let's define a couple of terms first. A *move* is defined to be a local disturbance of the solution (configuration), in which two PEs are involved (refer to Section 4.2 for examples). Note that a *perturbation* may result from more than one *move* by PEs, i.e., from a set of concurrent moves.

In SMC PSA, local moves at PEs which are involved in a perturbation form one global perturbation of configuration. The difference in the cost due to the configuration change is evaluated by all PEs cooperatively in parallel. One PE, the *scheduler*, takes care of the perturbation control, collection of local evaluation results, decision, and scheduling of the annealing.

In this section, three variations of SMC PSAs similar to the conventional PSA schemes are described. Those will be compared with the proposed MMC PSA. They are the *single move scheme*, the *multiple move scheme*, and the *hybrid move scheme*. Each of these variations has a different perturbation strategy.

2.2.1 Single Move Scheme

In the single move scheme, a perturbation results from a move. The scheduler PE randomly chooses one pair of PEs which are to interact with each other for perturbation. The primary PE of the two initiates the perturbation. For the perturbed configuration, the partial cost evaluated by each PE is collected at the scheduler. Thus, a perturbation is done by a pair of PEs while the evaluation is carried out by all PEs. Decision on the perturbation is made by the scheduler and is broadcast to all other PEs.

This scheme allows only one move between a PE pair per perturbation, allowing an elaborate annealing to be carried out. As a consequence, especially in the final stage of the annealing, this scheme could carefully proceed to a (near) optimal solution due to gradual change in the configuration. However, as shown in Fig. 2a, most PEs are idle when the the selected PE pair is working on a perturbation. This degrades the parallelization efficiency.

2.2.2 Multiple Move Scheme

In the multiple move scheme, all PEs are paired up and they communicate with each other when performing moves. Primary PEs initiate a perturbation by selecting moves; there can be $N/2$ moves per perturbation as shown in Fig. 2b. Then, each perturbed configuration is evaluated by the corresponding PEs and the local results are collected at the scheduler, which is in charge of the overall annealing scheduling. A decision is made and broadcast by the scheduler to update the intermediate configuration in the same way as in the single move scheme.

The multiple move scheme can change the cost more rapidly than the single move scheme since the configuration change per perturbation can be made larger. However, near the final stage of annealing, the solution (cost) being trapped in a local optimum may easily oscillate around a certain configuration and not converge toward a global optimum. In this case, it would be desirable to modify cost by a small amount. Also, since the decision is still made by one PE (scheduler) while the others are idle, a substantial degradation of efficiency may result in, especially when the number of PEs is large.

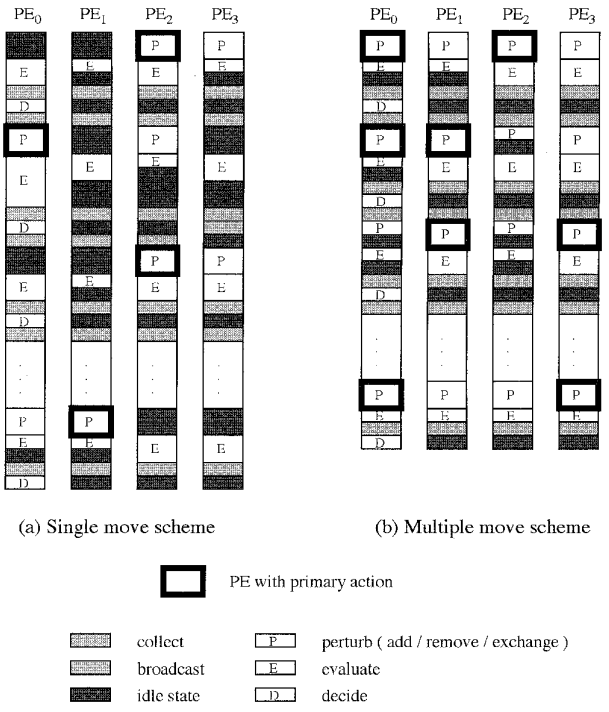


Fig. 2. SMC PSA schemes (a) single move scheme, (b) multiple move scheme.

2.2.3 Hybrid Move Scheme

The hybrid move scheme takes advantage of both the single move and the multiple move schemes. The basic idea appears to be equivalent to employing a range limiting function which, in general, restricts the range of moves according to the temperature [3].

In the hybrid move scheme, the *acceptance rate* (the number of perturbations accepted/the number of perturbations tried), is measured during annealing. It is used to adaptively determine the type of perturbation to be performed. The single move would be desirable in order to generate a small perturbation when the acceptance rate is low. At high acceptance rates, multiple moves to generate a large perturbation may lead to a fast reduction of the cost. Utilizing the acceptance rate does not add a heavy burden to the scheduler because the information at each PE can be delivered to the scheduler during the information exchange preceding a decision.

2.3 Multiple Markov Chain PSA

The approach of SMC PSA has at least two fundamental drawbacks. First, parallelism is limited since it is confined to a single search path. Second, a very high overhead is to be paid due to too frequent communication.

In order to improve performance of SMC PSA, one may relax the restriction that all PEs follow a single search path. One possibility is to have PEs trace multiple search paths at the same time, i.e., multiple Markov chain (MMC) PSA. Every PE has the entire search space, and the perturbation, evaluation and decision are performed by each PE individually. At the end of the algorithm, the best solution among the PEs is selected. The initial temperature, the

cooling coefficient and the equilibrium conditions can be independently decided and controlled by each PE. Since each PE randomly perturbs a configuration independently of others, it would be highly unlikely that any two PEs follows the same path. For better performance, PEs may be allowed to interact with each other as will be described later.

In order to guarantee a reasonable speed-up, the number of trials at a temperature is reduced by a certain factor, called *reduction factor* (refer to Section 3 for more details). Then, it is shown that we can still get an equivalent or even better solution.

Since PEs in the MMC PSA do not have to interact as frequently as in the SMC schemes, communication overhead will be significantly lower, which leads to a higher speed-up and efficiency. Also, the system is less likely to be trapped in a local optimum because multiple search paths are traced simultaneously.

Three schemes, namely, the *noninteracting scheme*, the *periodic exchange scheme*, and the *dynamic exchange scheme*, will be described below.

2.3.1 Noninteracting Scheme

Each PE independently perturbs the configuration, evaluates the cost, and decides on the perturbation. PEs do not interact during individual annealing processes until all PEs find their final solutions. Then the best of the solutions is saved and the others are discarded. One potential problem is that, as illustrated in Fig. 3a, all other PEs may have to wait for the PE with the longest search path (chain), resulting in idle time in every PE (except one), which may degrade the efficiency substantially.

In typical combinatorial problems, the computational load for perturbation and evaluation varies with configuration. Thus, as the annealing proceeds, the variance of the accumulated computational load among PEs becomes larger due to the random nature of SA, i.e., a larger variance in computation (or idle) time. Because PEs do not interact with each other until the end of SA, some of them may not perform useful computations. An efficient information exchange among PEs may be able to prevent PEs from performing unproductive computation or being idle so that the efficiency of parallelization can be improved.

2.3.2 Periodic Exchange Scheme

One possible way to improve the performance, referred to as the *periodic exchange scheme*, is illustrated in Fig. 3b. In this scheme, PEs exchange local information including the intermediate solutions and their costs with a fixed period. Then, each PE restarts from the best of the intermediate ones. Alternatively, to reduce the chance to be trapped, a random selection among the equilibrium solutions may be employed [11]. Also, depending on the intermediate cost distribution, the population update used in the genetic algorithm can be adopted [20]. We define a Markov chain between two successive information exchanges as a *segment*. Note that the multiple Markov chains are not completely independent, but they are within each segment.

Compared to the noninteracting scheme, communication overhead in this periodic exchange scheme would be

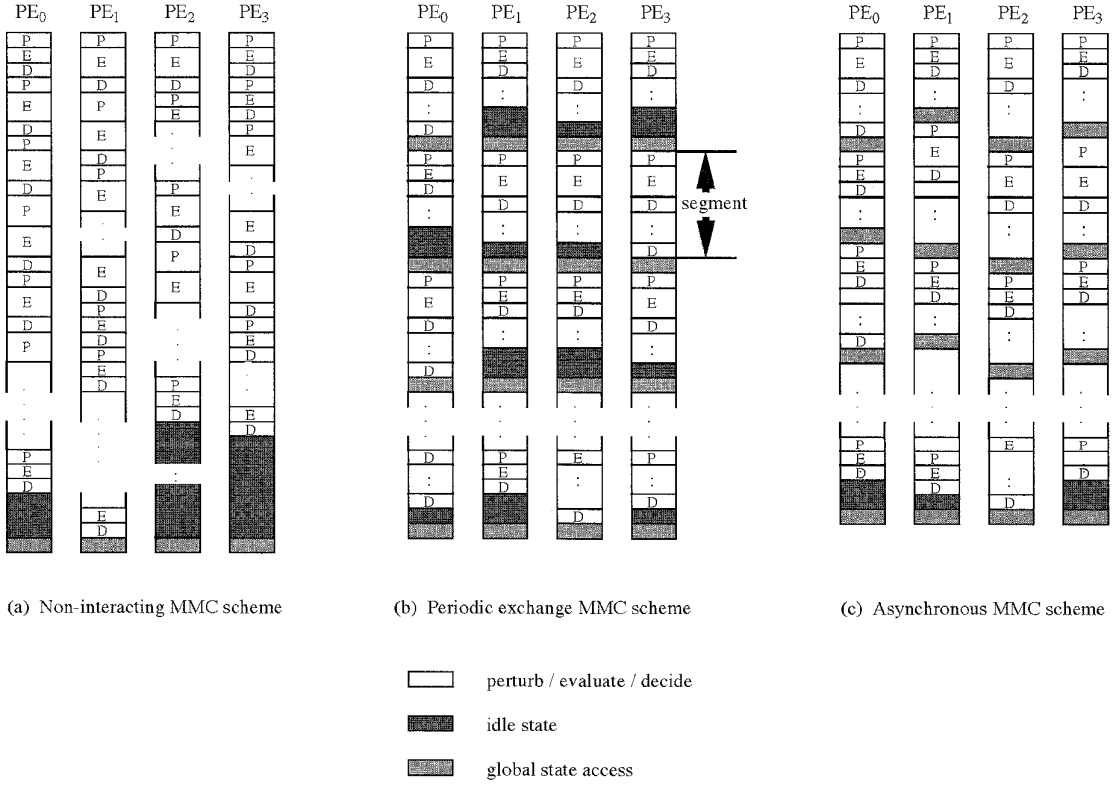


Fig. 3. MMC PSA schemes (a) noninteracting scheme, (b) periodic exchange scheme (synchronous), (c) asynchronous scheme.

increased. However, each PE can utilize the information from other PEs such that the decrease in computations and idle times can be greater than the increase in communication overhead. For instance, a certain PE which is trapped in an inferior solution can recognize its state by comparing it with others and may accelerate the annealing procedure. That is, PEs may collectively converge to a better solution. Nevertheless, the period of the information exchange needs to be carefully selected depending on how the intermediate solutions affect the convergence of other PEs.

This scheme resembles the *division algorithm* with communication proposed in the literature [7]. However, note that this scheme does not require communication among all PEs at every temperature while the division algorithm does. Communication at every temperature would incur too much synchronization overhead.

2.3.3 Dynamic Exchange Scheme

The statistical data obtained during execution may be utilized to adaptively control the SA process in each PE to further reduce the execution time. For example, the acceptance rate which is closely related to the annealing state can be utilized. The periodic exchanges may induce unnecessary and untimely information exchanges. Moreover, an intermediate solution derived at an insufficiently cooled state can hamper the convergence of other PEs.

We propose a new MMC PSA, the *dynamic exchange scheme*, which adaptively determines when information is to be exchanged. In the dynamic exchange scheme, whenever the following two conditions are satisfied, the information exchange among PEs is carried out. First, a certain period of time has to elapse, i.e., to allow each PE a sufficient independent annealing. Second, the acceptance rate is below a certain value, i.e., some PEs arrive at significantly better solutions. That is, PEs exchange information only when necessary rather than with a fixed period. In this way, PEs more efficiently guide each other to a higher quality solution. Also, communication time can be reduced substantially. In addition, to utilize the idle time, all PEs may continue (even after the termination criterion is met) their own annealing processes by exchanging their solutions until the PE, for which the termination criterion is met last, finds its final solution. In such a case, all PEs actually terminate their annealing simultaneously.

2.4 Asynchronous MMC PSA

The MMC PSAs described in the previous section are synchronous schemes in that all PEs should be ready before an information exchange takes place. While the synchronous MMC PSA is expected to find a global optimal or near optimal solution faster than the conventional SMC PSAs, there is still potential for considerable performance (especially speed) improvement. Each PE in the synchronous MMC PSA needs to wait for other PEs in each segment before it can communicate with them to get their current solutions (*local states*) for comparison as illustrated in Fig. 3b. Therefore, unless the segment length is the same for all paths or PEs (an extremely unlikely condition), efficiency of parallelization is degraded due to the idle time of each PE. As the variation in segment length among PEs increases, the efficiency decreases.

In order to further improve the performance of the MMC PSA, we propose asynchronous communication among PEs accessing the global state to reduce or eliminate the idle times. Each PE follows a separate search path, accesses the *global state* which consists of the current best solution and its cost whenever it finishes a segment, and updates the state if necessary. The global state is stored in a memory location that can be accessed by all PEs. Once a PE gets the global state, it proceeds to the next segment without any delay. For the asynchronous MMC PSA, the definition of *segment* needs to be generalized to a search path or Markov chain that a PE follows between two successive accesses (communications) to the global state. Note that this definition is valid for the synchronous MMC PSA also.

In the case of the asynchronous MMC PSA, each PE locks the global state before accessing it at the end of each segment, and unlocks it subsequently for other PEs. If the local state is better than the global state, it updates the global state by its local state. Otherwise, it updates its local state by the global state. Then, without any delay, it proceeds to the following segment. On the contrary, in the case of the synchronous MMC PSA, each PE accesses and updates the global state in the same manner, but needs to wait until all other PEs finish the updating in each segment. Then, the global state which is the best solution (and its cost) found in that segment is copied by all PEs before they initiate their next segments.

The asynchronous MMC PSA has the following features compared to the synchronous MMC PSA. First, no PE will be idle during annealing. That is, different lengths of segments do not cause idle states until a Markov chain ends since communication (global state access) is carried out asynchronously. Second, communication overhead itself is less in the asynchronous implementation than in the synchronous one. In the synchronous MMC PSA, local states are to be exchanged among PEs; during this time computations are not performed. In contrast, an isolated access to the global state is needed by each PE at the end of each segment in the asynchronous MMC PSA. Simultaneous accesses by PEs, which result in conflicts, are unlikely. Third, the probability of being trapped in a local optimum can be smaller depending on the cost function. This is mainly due to the fact that not all PEs start from the same state in each segment while they do in the synchronous MMC PSA.

3 Performance Analysis

In this section, we model the PSAs described in the previous section and compare their performance. The main purpose is to show analytically that the MMC PSA can perform better than the SMC PSA and that the asynchronous MMC PSA can be faster than the synchronous MMC PSA.

3.1 SMC PSA vs. MMC PSA

We consider the MMC PSA first. Let the time for one iteration (P/E/D) be denoted by t_u ($= t_p + t_e + t_d$) and I_{ik} denote the number of trials at the kth temperature (outer) loop of PE_i. Then the total execution time of the non-interacting

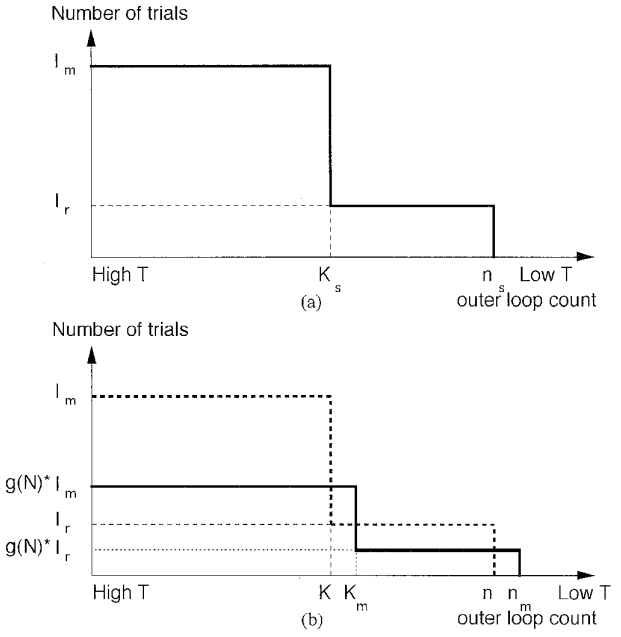


Fig. 4. Simplified annealing model (number of trials vs. outer loop count) for (a) SMC PSA and (b) MMC PSA.

MMC PSA with N PEs, $t_{mmc}(N)$ can be written as follows:

$$t_{mmc}(N) = \max_{i\in(1,\cdots,N)} \sum_{k=1}^{n_i} I_{ik}t_u + T_c, \tag{1}$$

where n_i is the final loop count of PE_i and T_c is the total communication time. For the interacting MMC PSAs, the first term becomes $\sum_{j=1}^{M} \max_{i\in(1,\cdots,N)} I_{ij}t_u$, where M is the number of segments and I_{ij} is the total number of trials in the jth segment of PE_i.

It is reasonable to divide the temperature into higher and lower temperature zones by the *cut-off temperature*. The cut-off temperature (T_K) is defined as a temperature at which the number of trials per temperature is noticeably reduced. Then, the annealing curve, the number of trials versus the outer loop count, can be approximated by a two-step curve as shown in Fig. 4. Note that the execution time is proportional to the area under the curve. The reduction factor, $g(N)$, which was introduced in Section 2.3, obviously affects the parallel execution time. A typical example of $g(N)$ is $\frac{1}{N}$. For the MMC PSA, the number of trials at a temperature is usually limited by $g(N)\cdot I_m$ from the initial temperature to the cut-off temperature and by $g(N)\ I_r$ from the cut-off temperature to the final temperature, where I_m and I_r are the maximum numbers of trials allowed at high and low temperature regions in the sequential SA, respectively.

Let's denote the temperature (outer) loop count at the cut-off temperature by K and the total outer loop count by n for the sequential SA. Then, the corresponding K_m and n_m for the MMC PSA can be represented as $K_m = \max_{i\in(1,\cdots,N)} K_i$ and $n_m = \max_{i\in(1,\cdots,N)} n_i$ where K_i and n_i are K and n of the PE_i, respectively. The outer loop counts at the final temperature (n_m) and the cut-off temperature (K_m) are random variables which vary with other parameters and the characteristics of cost distribution.

Let the times for perturbation, evaluation and decision be t_p, t_e, and t_d, respectively, and the scheduling time per iteration be t_s. Then, the execution time of a sequential SA, t_{seq}, is

$$t_{seg} = \left(I_m K + I_r(n-K)\right)\left(t_p + t_e + t_d\right) + nt_s. \tag{2}$$

Similarly, referring to Fig. 4b, the execution time for the MMC PSA, $t_{mmc}(N)$, can be represented from (1) as follows:

$$t_{mmc}(N) = g(N)\left(I_m K_m + I_r(n_m - K_m)\right)\left(t_p + t_e + t_d\right) + Mt_c \log_2 N. \tag{3}$$

In (3), it is assumed that the communication among N PEs can be completed in $\log_2 N$ steps (e.g., in a hypercube), and t_c is the communication time between adjacent PEs. We note that $M = 1$ for the noninteracting scheme, and $M = \left\lceil n_m / L_p \right\rceil$ for the periodic exchange scheme, where L_p is the number of temperature loops between adjacent global state accesses (refer to Fig. 3). M for the dynamic exchange scheme is smaller than that for the periodic exchange scheme.

From (2) and (3), the speed-up for the MMC PSA, S_{mmc}, can be derived as

$$S_{mmc} = \frac{t_{seq}}{t_{mmc}(N)} = \frac{C_0\left(t + t_d\right) + nt_s}{C_m\left(t + t_d\right)g(N) + Mt_c \log_2 N + n_m t_s} \tag{4}$$

where $t = t_p + t_e$, $C_0 = I_m K + I_r(n-K)$, and $C_m = I_m K_m + I_r(n_m - K_m)$.

For small t_s and t_c, if there is no significant difference between K and K_m, and n and n_m, the speed-up would be nearly $1/g(N) = N$ assuming $t \gg t_s, t_c$. However, as the unit communication time (t_c), the number of PEs (N) or the total number of segments (communication phases) (M), increases, the speed-up is degraded. Even if the two MMC PSA schemes with communication complete at the same outer loop count (n_m), the amount of communication required by the dynamic exchange scheme is less than that by the periodic exchange scheme.

Now, consider the SMC PSA. Suppose that the perturbation and evaluation steps are parallelized, but not the decision step. The execution time for the SMC PSA can be written as (refer to Figs. 2 and 4)

$$t_{smc}(N) = \left(I_m K_s + I_r(n_s - K_s)\right)\left(\frac{t_p + t_e}{N} + t_0 N + t_d + f_c t_c' \log_2 N\right) + n_s t_s. \tag{5}$$

In (5), t_c' is the unit communication time of the SMC PSA which corresponds to the t_c of the MMC PSA, f_c is the number of communications per iteration, and t_o is the parallelization overhead for data management (e.g., the scheduler PE prepares data for parallel perturbation and evaluation, and the information to be broadcast to all PEs; refer to Section 2.2). t_c

and t_c' depend on data size, communication algorithm and computer system, and t_o depends on the perturbation and evaluation strategies.

Then, the speed-up of SMC PSA is derived as follows:

$$S_{smc} = \frac{t_{seq}}{t_{smc}(N)} = \frac{C_0(t + t_d) + nt_s}{C_s\left(\frac{t}{N} + t_o N + t_d + f_c t_c' \log_2 N\right) + n_s t_s} \quad (6)$$

where $C_s = I_m K_s + I_r(n_s - K_s)$.

To compare the two strategies (SMC and MMC),

$$\frac{S_{mmc}}{S_{smc}} = \frac{t_{smc_N}}{t_{mmc}(N)} = \frac{C_s\left(\frac{t}{N} + t_o N + t_d + f_c t_c' \log_2 N\right) + n_s t_s}{C_m(t + t_d)g(N) + Mt_c \log_2 N + n_m t_s}. \quad (7)$$

Even though the size of data exchanged at a time in the SMC PSA may be smaller than that in the MMC PSA $(t_c' \le t_c)$, the total communication time of the SMC PSA is much larger than that of the MMC PSA due to the fact that PEs in the SMC PSA communicate much more frequently, i.e., $C_s f_c t_c' \log_2 N \gg Mt_c \log_2 N$ since $C_s f_c \gg M$. Under the assumptions that $t_s, t_o \ll t, C_s \approx C_m$ and $g(N) = 1/N, \frac{S_{mmc}}{S_{smc}} > 1$, i.e., the MMC PSA is faster than the SMC PSA. The assumptions are valid in most SA problems since

1) t_p and t_e are usually much larger than the scheduling time, t_s,
2) the total number of trials in the MMC PSA can be made similar to that in the SMC PSA and
3) the reduction factor $g(N) = 1/N$ is applicable because the sufficient cooling can lead PEs to achieve a quasi-equilibrium in the proposed interacting MMC PSA.

Even if a less drastic reduction factor like $g(N) = \frac{1}{\sqrt{N}}$ is employed (then more trials are made, therefore, a better solution can be obtained), $S_{mmc} > S_{smc}$ is usually satisfied under reasonable conditions such as $t : t_o : t_c' = 10 : 1 : 1$. Only in case that $t \gg f_c t_c' \log_2 N, S_{smc}$ can approach S_{mmc}.

It would be interesting to see how problem-dependent the performance (speed-up) of each strategy is. For this purpose, one may employ the derivative of speed-up with respect to t where $t = t_p + t_e$, i.e., $\partial S_{mmc} / \partial t$ and $\partial S_{smc} / \partial t$. Note that a different problem would have a different value of t.

From (4) and (6), we can derive the following derivatives when $t_s, t_o, t_d \ll t$, and $g(N) = 1/N$,

$$\frac{\partial S_{mmc}}{\partial t} \approx \frac{\frac{C_0}{C_m^2}\left(Mt_c \log_2 N\right) + n_m t_s}{\left(\frac{t}{N} + \frac{Mt_c \log_2 N + n_m t_s}{C_m}\right)^2} \quad (8)$$

and

$$\frac{\partial S_{smc}}{\partial t} \approx \frac{\frac{C_0}{C_s}\left(t_o N + f_c t_c' \log_2 N + \left(1 - \frac{1}{N}\right)t_d\right)}{\left(\frac{t}{N} + \frac{n_s t_s}{C_s}\right)} \quad (9)$$

respectively.

If $C_s \approx C_m$ and $t_c, t_c' \ll t$, the second term of the denominator in (8) is much smaller than the first term due to $C_m t \gg Nn_m t_s$. Also, the second term of the denominator in (9) is much smaller than the first term due to $C_s t \gg Nn_s t_s$. Comparing the numerators of (8) and (9), $C_m > \frac{M}{f_c}$ and $C_m t_0 > \frac{n_m}{N} t_s$ are satisfied in most applications. Therefore, the following inequality holds in most cases:

$$\frac{\partial S_{smc}}{\partial t} > \frac{\partial S_{mmc}}{\partial t}. \quad (10)$$

That is, the performance (speed-up) of MMC PSA is more consistent than that of SMC PSA as the optimization problem varies. This is mainly due to the fact that the MMC PSA parallelizes the whole annealing steps while the SMC PSA does only a part which can vary significantly with problem.

3.2 Synchronous MMC PSA vs. Asynchronous MMC PSA

The length of a segment can be formulated as a *random variable* which has a certain distribution characterized by a probability density function (*pdf*). In the following analysis, it will be assumed that the random variable for segment length has the same distribution for all PEs and at all temperatures, given a problem and a scheduling scheme. This assumption is mainly for simplicity of analysis. Later, whenever necessary and possible, relaxing the assumption or the effect of deviation from the assumption will be discussed.

Let random variable l_{ik} denote *segment length* for the kth segment of PE_i .The unit of the segment length is time and is always non-negative. The mean and standard deviation of l_{ik} are μ_{ik} and σ_{ik}, respectively. Under the above assumption $\mu_{ik} = \mu$ and $\sigma_{ik} = \sigma$ for all i, k.

Let the total execution times of the asynchronous and synchronous MMC PSAs be t_{asyn} and t_{syn}, respectively. Suppose that N PEs are employed and each PE follows M segments. In the case of the synchronous MMC PSA, all PEs are synchronized at the end of each segment. Therefore, t_{syn} can be written as follows:

$$t_{syn} = \sum_{k=1}^{M} \max_{i \in (1,\ldots,N)} l_{ik}. \quad (11)$$

In the case of the asynchronous MMC PSA, each PE proceeds without synchronization until the end of the entire simulated annealing when local solutions are compared. Hence, t_{asyn} can be represented as follows:

$$t_{asyn} = \max_{i \in (1,\ldots,N)} \left(\sum_{k=1}^{M} l_{ik}\right). \quad (12)$$

From (11) and (12), one can see that t_{syn} can never be smaller than t_{asyn}.They become equal when the outcome of l_{ik} doesn't vary with i, which is practically impossible. For a more quantitative comparison, let l_{max} denote the maximum possible outcome (or upper bound) of l_{ik}. It is reasonable to assume that l_{max} depends on μ and σ. Noting that l_{max} is larger for a larger σ, given a class of underlying distribution, one may write

$$l_{max} = \mu(1 + h(\bar{\sigma})), \quad (13)$$

where $h(\cdot)$ is a monotonically increasing positive function

indicating how much l_{max} is deviated from μ, and $\overline{\sigma}$ is the standard deviation *normalized* by the mean, i.e., $\frac{\sigma}{\mu}$.

One may try to analytically derive l_{max} by assuming a specific *pdf* for l_{ik} .However, it may not be worthwhile due to the following reasons. First, it is almost impossible to derive an easily appreciable closed form formula describing maximum outcome for samples of the sum of random variables, t_{asyn}, though it is relatively easy for samples of a random variable. Second, it may not be feasible to build a probabilistic model, e.g., *pdf*, that precisely fits a given practical problem. Third, in most cases, a performance measure derived with the first and second order moments, i.e., mean and standard deviation, is sufficiently accurate and meaningful.

From (11),

$$t_{syn} = Ml_{\max} = \mu\left(1+h(\overline{\sigma})\right)M. \tag{14}$$

The standard deviation of $l_i = \sum_{k=1}^{M} l_{ik}$, which is the total segment length of PE_i in the asynchronous MMC PSA, is $\sigma\sqrt{M(M-1)\rho+M}$ where ρ is the correlation coefficient between segments l_{ij} and l_{il} for $j \neq l$, and $0 \leq \rho \leq 1$. After normalization by its mean which is μM, it becomes $\overline{\sigma}\sqrt{\frac{(M-1)\rho+1}{M}}$. With this result, (12) can be expressed as

$$t_{asyn} = \mu M\left(1 + h\left(\overline{\sigma}\sqrt{\frac{(M-1)\rho+1}{M}}\right)\right). \tag{15}$$

Now, denote the ratio of t_{syn} to t_{asyn} by r, which indicates the relative improvement of the execution time (it will be simply referred to as the *improvement factor*) of asynchronous MMC PSA over synchronous MMC PSA. It needs to be emphasized that r is not the speed-up (S_{mmc}) of asynchronous MMC PSA over *sequential* SA. From (14) and (15),

$$r = \frac{t_{syn}}{t_{asyn}} = \frac{1+h(\overline{\sigma})}{1+h\left(\overline{\sigma}\sqrt{\frac{(M-1)\rho+1}{M}}\right)} \tag{16}$$

where $M \geq 1$.

It is clear from (16) that $r \geq 1$ since $0 \leq \rho \leq 1$ and $h(\cdot)$ is monotonically increasing, i.e., the asynchronous MMC PSA is faster than the synchronous MMC PSA unless $\rho = 1$ for which $r = 1$. Though (16) is based on the assumptions mentioned above, several meaningful observations can be drawn from the equation.

3.2.1 Global State Access Frequency

First, consider the effect of M on r. Though M is the total number of global state accesses (or segments), it may be referred to as *global state access frequency* to indicate how often a PE accesses the global state, which should be linearly proportional to M. The global state access frequency, M, is one of the major parameters which differentiates the synchronous and asynchronous schemes. As M increases, i.e., PEs access the global state more frequently, PEs will be idle for a longer period of time in both PSAs. However, the idle time in the asynchronous scheme, which occurs only in the final segment, is relatively smaller than that in the synchronous scheme. This is due to the property that the normalized standard deviation $(\overline{\sigma})$ of a sum of random variables cannot be larger than the sum of the normalized standard deviations of individual random variables, as derived earlier. This difference becomes larger as M increases as can be seen in (16).

From (16), it is easy to see that r increases first and tends to saturate as the global state access frequency increases. This benefit of using the asynchronous MMC PSA is greater when the normalized standard deviation is larger. As the correlation between segments decreases, r becomes more advantageous to use the asynchronous scheme. The improvement factor r is maximized, given $\overline{\sigma}$ and M when $\rho = 0$ (segments are uncorrelated).

3.2.2 Number of Markov Chains

In the above derivation and discussion, the number of Markov chains is assumed to have no effect on l_{max} which influences the improvement factor. Theoretically, the maximum value of a random variable is fixed given an underlying distribution. When sampling the random variable in practice, the maximum of the samples or outcomes may not be the same as the theoretical one. The sample maximum is expected to approach the theoretical one as the number of samples increases. This practical aspect was not taken into account when l_{max} was formulated (refer to (13)). To make the improvement factor formula more realistic, (13) may be modified as follows:

$$l_{max} = \mu\left(1 + h(\overline{\sigma})f(N)\right) \tag{17}$$

where $f(N)$ is a monotonically increasing function of $N > 1$, $0 < f(2) < 1$, and $f(N)$ approaches one as N increases.

$f(N)$ can be considered as a calibrating function which indicates the *actual* effect of $h(\overline{\sigma})$ or $\overline{\sigma}$ on l_{max} as a function of N. As the number of PEs, N, increases, the improvement factor r (of the asynchronous MMC PSA over the synchronous MMC PSA) increases, but eventually saturates.

3.2.3 Communication Overhead

Another advantage of employing the asynchronous MMC PSA is that communication time itself can be saved (though it is not included in the equations of this section). In the synchronous MMC PSA implemented on a message passing multiprocessor, the local states are to be collected at a certain PE which will then derive the global state from them and broadcast it back to all other PEs. This data sharing would require communication time proportional to $D\log_2 N$ on a hypercube, for example, where D is the size of data (global state) to be exchanged and N the number of PEs [18]. If the synchronous MMC PSA is implemented on a shared memory multiprocessor, the communication time would be $O(DN)$ assuming that there is only one copy of the global state on the shared memory.

In the asynchronous MMC PSA, each PE just needs to access and update, if necessary, the global state independent of other PEs. Therefore, the communication time would be $O(D)$ on a shared memory system assuming that there is no conflict among PEs in the global state access. Of course, in the worst case when all PEs try to access the global state at the same time, the complexity would be $O(DN)$. However, this worst case is very unlikely to occur. On a message

passing multiprocessor, each PE may send its local state to a controller at the end of each segment. Then, the controller returns the global state (after updating it by the local state if necessary). The communication time complexity is the same as on a shared memory system. Therefore, the saving in communication time by the asynchronous scheme can be substantial when N is large.

3.2.4 Solution Quality

Any faster scheme which degrades solution quality significantly would not be acceptable unless speed is more important than solution quality. All PEs in the synchronous MMC PSA initiate each segment from the same global state (solution). However, a different PE in the asynchronous MMC PSA may see a different global state depending on the order in which PEs access the global state.

Suppose that a PE accesses the global state at the end of each segment. If a local solution (state) of the PE is better than the global solution, the PE updates the global solution by its solution. If not, the PE discards its solution and initiates next segment from the global solution (state). For example, suppose the PE with the best solution happens to update the global state *first* in a certain segment. Then, all PEs would start from the same global state at least in that segment. Unless this happens, some PEs would have initial solutions for a segment which are different from others. As in the genetic algorithm [20], this will reduce the probability that the global state is trapped in a local optimum. Therefore, the solution quality is not expected to be worse than that by the synchronous MMC PSA.

3.2.5 Problem Dependency

The cost function may vary with the problem. If the cost function, more specifically density or distribution of local optima, changes, it is likely for the pdf of l_{ik} to change resulting in a different σ. For a larger $\overline{\sigma}$, a higher r would be obtained as can be seen from (16).

4 Application to Graph Partitioning Problem

In order to experimentally examine the performance of the proposed MMC PSAs (synchronous and asynchronous), we have applied them to graph partitioning, which is NP-complete. Many engineering problems can be formulated as a graph partitioning problem [21], [22], [23], [24].

4.1 Problem Formulation

An *undirected* graph $G = (V, E)$ consists of a finite set of nodes (vertices) V and a finite set of edges (links) E, where an element of E is an *unordered* pair of nodes u and v in V, i.e., an edge in E is represented as $e_{u,v} = (u, v) \in E$, where $u, v \in V$. A *subgraph* (W, F) can be defined such that $W \subseteq V$ and $F \subseteq E$. If nonempty sets V_1 and V_2 are subsets of V such that $V_1 \cup V_2 = V$ and $V_1 \cap V_2 = \varnothing$, $\{V_1, V_2\}$ is said to be a *partition*. *Cut* is a set of edges the removal of which results in a partition. The weight of a node u is denoted by $\omega_v(u)$ and the weight of an edge (u, v) by $\omega_e((u, v))$.

4.1.1 Graph Partitioning Problem

Given an undirected graph $G = (V, E)$ with node and edge weights, $(\omega_v(\cdot), \omega_e(\cdot))$, where V is a set of nodes and E is a set of edges, find a partition of V into R disjoint sets $\{V_1, V_2, \ldots, V_R\}$ such that the cost function below is minimized.

The cost function, which is to be minimized, is chosen as

$$C(\{V_1, \ldots, V_R\}) = aM(\{V_1, \ldots, V_R\})^x + bL(\{V_1, \ldots, V_R\})^y \quad (18)$$

where

$$M(\{V_1, \ldots, V_R\}) = max\{\|V_1\|, \ldots, \|V_R\|\},$$

and

$$L(\{V_1, \ldots, V_R\}) = \sum_{(u,v) \in \delta(V_i, V_j), 1 \le i, j \le R, i \ne j} \omega_e((u, v)),$$

where $\|V_i\| = \sum_{v_n \in V_i} \omega_v(v_n)$ and $\delta(V_i, V_j)$ is the cut of the sets V_i and V_j, which is defined as

$$\delta(V_i, V_j) = \{(u, v) \in E \mid u \in V_i \wedge v \in V_j\}.$$

Suppose that $a = b = x = y = 1$ with $\omega_v(\cdot) = 1$ and $\omega_e(\cdot) = 1$ for all nodes and edges in the graph. Then, $C(\cdot) = M + L$, where M is the largest number of nodes in a subgraph and L is the total number of edges between subgraphs. In this case, minimizing $C(\cdot)$ requires balancing the node distribution among subgraphs and at the same time reducing the number of edges lying between subgraphs. By choosing an appropriate set of values for the parameters (a, b, x, y), one may formulate a cost function suitable for a given application. Also, it needs to be pointed out that our approach of MMC PSA is not limited to graph partitioning only, but applicable to any optimization problems.

4.2 Perturbation Algorithm

In SA, the next candidate configuration (partition) is determined by a perturbation scheme. In the graph partitioning problem, we define three ways of perturbation, i.e., *add, remove, and exchange*, (each of which is referred to as a *move*), where an *add* in a subgraph corresponds to a *remove* in the paired subgraph. Note that a perturbation may result from multiple moves when there are more than two partitions (subgraphs). One of the moves is selected at random according to the node distribution. The add, remove and exchange are uniformly generated except in the extreme cases (e.g., the difference in the number of nodes exceeds a certain constant or the selected subgraphs have only non-removable nodes). Fig. 5 shows the perturbation examples where (a) is the initial configuration and the resulting configurations after a move—(b) add, (c) remove and (d) exchange—in view of the subgraph A (the *primary* subgraph).

The perturbation scheme, which is referred to as the *heuristic rearrangement algorithm*, is incorporated into the inner loop to efficiently perturb the configuration. In the heuristic rearrangement algorithm, the nodes in the primary subgraph (which is randomly chosen among all subgraphs) are sorted periodically (not in every iteration) in the ascending order according to internal degree (where the internal degree, $deg(v_i)$, is defined as the sum of weights of

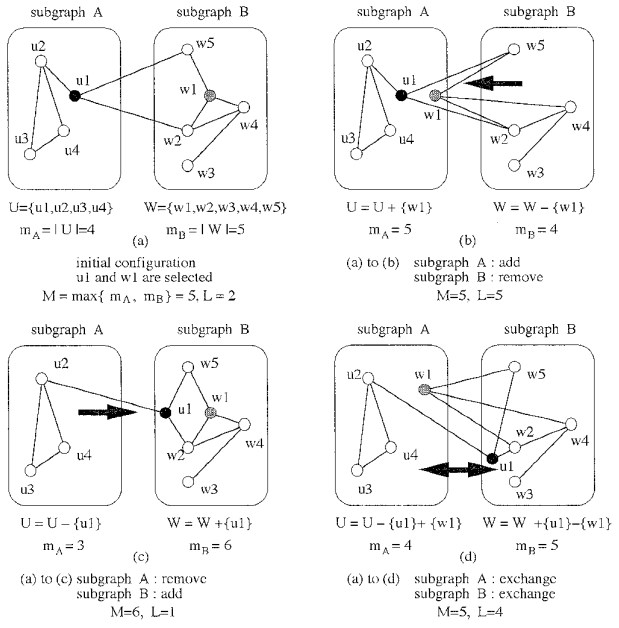


Fig. 5. Perturbation examples (a) initial configuration, (b) add, (c) remove, (d) exchange.

internal edges incident on the node v_i, where an internal edge is an edge that does not lie between subgraphs). The top node in the sorted list is involved in the move immediately following the sorting. Selection of nodes to be moved or exchanged is basically random, but is designed such that no node is completely excluded.

This heuristic rearrangement algorithm generates the next configuration more effectively than a simple random perturbation since it can reduce the unnecessary perturbations to a great extent by utilizing the connectivity information of subgraph nodes.

4.3 Communication Algorithms

Information exchange among PEs is necessary for both types of PSAs described earlier. In the SMC PSA, communication is indispensable when subgraphs are perturbed, when the locally evaluated costs at different PEs are collected at a certain PE, and when the decision is broadcast. In the MMC PSA, local information needs to be exchanged to update the global state (the current configuration and its cost) when each PE wants to check other PEs status during annealing and when all PEs finish annealing. Thus, three different types of communications are needed:

1) broadcast,
2) collect, and
3) exchange.

These communications can be efficiently implemented using the *pseudobinary tree* embedded in the hypercube [18].

4.4 Results and Discussions

The proposed PSA algorithms have been applied to the IEEE 30-node and 118-node standard networks, labeled as G30 and G118, respectively, a randomly generated 200 node weighted graph (G200), where the range of node weight (10, 20) and the range of edge weight (1, 5) and 1,200 node graph (G1200). Their performance has been evaluated under the same conditions (parameters listed in Section 4.4.1). All performance figures, cost and execution time, are the averaged (not best) ones.

The PSA schemes have been compared based on the minimum cost found and the parallel execution time. In order to evaluate the convergence property of the proposed schemes, the best results obtained by the sequential algorithm of Irving and Sterling. [21] are referred to, which was reported to find significantly better solutions than an iterative improvement algorithm. The performance results of the proposed schemes for the cost function in (18) ((a, b, x, y) are usually set to (1, 1, 2, 3) as in [21]) are provided for G30 (average degree: 2.73), G118 (average degree: 3.05), G200 (average degree: 2.77), and G1200 (average degree: 2.98).

For further comparison, another SMC PSA scheme proposed by Roussel-Ragot and Dreyfus. [11], to be referred to as Modified R-R (MRR), has been modified for better adaptation to the graph partitioning problem and implemented. It utilizes the acceptance rate in determining next solution. In the MRR, PEs asynchronously continue annealing until at least one PE finds an acceptable solution. Then, one of the accepted solutions is randomly selected when the acceptance rate is high ($> 1/N$) while the best solution is chosen when the acceptance rate is low ($< 1/N$) [11]. Note that PEs have to communicate with each other frequently especially when the temperature is high.

4.4.1 Test Environment and Parameters

A graph is initially partitioned at random. The initial temperature (T_i) for each PE is not predetermined identically, but is independently derived in the beginning with randomly selected sample configurations. In addition, the standard deviation of the intermediate solutions at each PE is calculated at every temperature to determine the next temperature [19]. In the following, the annealing parameters for the graph partitioning problem, which are empirically determined, are summarized:

- maximum number of temperature decrement : 250
- maximum number of iterations at a temperature (I_m): $k \cdot 100 \cdot (R-1) \cdot g(N)$ ($k = 1$ for G30, $k = 2$ for G118, $k = 3$ for G200, $k = 10$ for G1200 where $g(N)$ is usually set to $1/N$.)
- minimum number of iterations at a temperature (I_r) : $max(0.3\ I_m, 5)$
- static temperature decrement factor α (cooling rate): 0.95
- threshold acceptance rate for hybrid move : 0.5

The synchronous MMC PSA schemes have been compared with the SMC PSA schemes on the *Intel iPSC/2*, a message-passing system. The synchronous and asynchronous MMC PSAs have been tested on a *shared-memory* multiprocessor system, *BBN GP1000*. The typical implementation results are tabulated in Tables 1 through 6 where cost and speed-up are the averaged ones for multiple executions in each case.

4.4.2 SMC PSA

Among the SMC PSA schemes, the hybrid move scheme has found the best configuration in all cases tested, as

TABLE 1
PERFORMANCE OF PSA SCHEMES FOR IEEE 30-NODE NETWORK (G30), $(x, y) = (2, 3)$

		(a, b) = (1, 3)						(a, b) = (10, 1)					
		N = 2, R = 2			N = 4, R = 4			N = 4, R = 4			N = 8, R = 8		
PSA/SSA scheme		cost	T[ms]	S	cost	T[ms]	S	cost	T[ms]	S	cost	T[ms]	S
SMC PSA	Single Move	600	17948	0.5	1657	10731	1.4	1539	8251	1.5	3010	6668	3.6
	Multiple Move	—	—	—	2268	7625	2.0	1810	7880	1.6	3113	5854	4.1
	Hybrid Move	—	—	—	1617	6556	2.3	1322	5861	2.1	2687	5881	4.0
	MRR PSA	633	18774	0.5	1657	9742	1.6	1539	8240	1.5	3104	7743	3.1
MMC PSA	Static Noninter.	553	5362	1.7	1636	4359	3.5	1322	3909	3.2	2675	3459	6.9
	Periodic Exch.	600	4755	1.9	1657	4174	3.6	1322	3507	3.5	3145	3201	7.5
	Dynamic Exch.	516	4831	1.9	1225	4283	3.5	1322	3451	3.6	2687	3097	7.8
SSA	N=1 Heuristic	592	9082	1.0	1225	15190	1.0	1322	12360	1.0	2687	24005	1.0
	Irving	649	n/a	—	1617	n/a	—	1322	n/a	—	n/a	n/a	—

(—) In the SMC, the multiple move and hybrid move schemes are the same as the single move scheme for $N = R = 2$.

TABLE 2
PERFORMANCE OF PSA SCHEMES FOR G30, G118, AND G1200

	SMC PSA						MMC PSA					
	Hybrid Move			Modified R-R.			Noninteracting			Dynamic Exchange		
N	cost	T[ms]	S	cost	T[ms]	S	cost	T[ms]	S	cost	T[ms]	S
1	340	15910	1	—	—	—	340	15910	1	—	—	—
2	442	12240	1.3	340	26520	0.6	320	6311	1.6	320	4482	1.7
4	396	6696	2.4	571	29703	0.5	365	3863	3.2	330	3153	3.3
8	370	2864	5.6	768	14451	1.1	338	2482	6.4	320	1974	6.5
16	487	2796	5.7	391	12757	1.2	320	2098	7.6	320	1302	12.2

(a) G30, R=2, (a,b)=(1,1), (x,y)=(2,3)

	SMC PSA						MMC PSA					
	Hybrid Move			Modified R-R.			Noninteracting			Dynamic Exchange		
N	cost	T[ms]	S	cost	T[ms]	S	cost	T[ms]	S	cost	T[ms]	S
1	4812	33159	1	—	—	—	4812	33159	1	—	—	—
2	5230	19804	1.7	6885	32460	1.0	4417	17047	1.9	3824	15736	2.1
4	6616	10738	3.0	8058	21717	1.5	4712	9584	3.5	4210	8880	3.7
8	8436	7712	4.3	14191	29417	1.2	6857	4901	6.8	4210	4510	7.3
16	8614	6109	5.4	9659	23169	1.4	5231	2883	11.5	4481	2429	13.7

(b) G118, R=2, (a,b)=(1,1), (x,y)=(2,3)

	SMC PSA						MMC PSA					
	Hybrid Move			Modified R-R.			Noninteracting			Dynamic Exchange		
N	cost	T[s]	S	cost	T[s]	S	cost	T[s]	S	cost	T[s]	S
1	1.27e6	3219	1	—	—	—	1.27e6	3219	1	—	—	—
2	1.33e6	2118	1.5	1.73e6	2660	1.2	1.13e6	1462	2.2	1.13e6	1461	2.2
4	2.02e6	999	3.2	3.28e6	1778	1.8	1.50e6	727	4.4	1.37e6	704	4.6
8	3.24e6	787	4.1	3.62e6	1504	2.1	1.83e6	382	8.4	1.11e6	338	9.5
16	4.57e6	733	4.4	4.96e6	1668	1.9	2.67e6	239	13.4	1.83e6	213	15.1

(c) G1200, R=2 (a,b)=(1, 1), (x,y)=(2,3)

shown in Tables 1 and 3. In terms of the execution time, the multiple move scheme takes the least amount of time, and the single move scheme takes the longest since it changes the graph configuration gradually. While the multiple move scheme can rapidly perform the initial stages of annealing, it is susceptible to local minima and oscillations in the final stages. The single move is better at the final stage of the annealing. This is the reason why the hybrid move scheme performs best among the SMC PSA schemes.

TABLE 3
PERFORMANCE OF PSA SCHEMES FOR IEEE 118-NODE NETWORK (G118), $(x, y) = (2, 3)$

(a,b)=(1, 1)		N = 2, R = 2			N = 4, R = 4			N = 8, R = 8			N = 16, R = 16		
PSA/SSA scheme		cost	T[ms]	S	cost	T[ms]	S	cost	T[ms]	S	cost	T[ms]	S
SMC PSA	Single Move	5230	19804	1.2	7526	33471	1.3	27625	31350	3.1	175841	38693	4.6
	Multiple Move	—	—	—	9369	30501	1.4	30416	23138	4.2	314657	23803	7.4
	Hybrid Move	—	—	—	6856	32837	1.3	24965	26265	3.7	103967	29917	5.9
	MRR PSA	6885	32460	0.8	12017	40579	1.0	64529	35597	2.7	175785	43277	4.1
MMC PSA	Static Noninter.	4417	17047	1.5	4819	12849	3.3	14353	17669	5.5	85380	15967	11.1
	Periodic Exch.	4283	16084	1.5	4188	10946	3.8	14012	14166	6.9	90975	12015	14.8
	Dynamic Exch.	3824	15736	1.6	3961	10640	3.9	11489	13648	7.1	74232	11184	15.8
Seq. Heuristic SA (N=1)		5356	24725	1.0	4265	41924	1.0	14353	97180	1.0	85409	177233	1.0

The convergence of the MRR (SMC) PSA seems to be poor since the random selection rule (which is designed to preserve the property of the sequential SA) at high temperatures may not be suitable for the graph partitioning problem. The solution quality is usually worse than those by our SMC PSA schemes as can be seen in Tables 1, 2, and 3. Also, the speed-up is noticeably lower in the MRR PSA than in our SMC PSA schemes especially for a large N. In the MRR PSA, a larger amount of information (perturbation, evaluation and decision results) needs to be exchanged at every step among PEs to generate N different states.

It is observed that the SMC PSA sometimes performs even worse than the sequential version especially for a small size problem (Table 1). For a small size of graph, the computational load in the perturbation and evaluation steps (which are to be parallelized) is relatively small compared to the total execution time, and the relative communication overhead is larger. That is, the overhead can offset the speed gain from the limited exploitation of parallelism in computation. For the MRR PSA, in which even evaluation is not parallelized, the performance degradation is more apparent.

The single Markov chain approach is shown not to be suitable for efficient parallelization of the SA algorithm particularly for the graph partitioning problem, since the perturbation and evaluation of the configuration (for which the workload is shared by PEs) do not occupy a large portion of the execution time. In addition, the communication overhead involved in each perturbation and decision can be overwhelming.

4.4.3 Synchronous MMC PSA

Among the synchronous MMC PSAs, the proposed dynamic exchange scheme, which utilizes the acceptance rate in determining when to communicate, usually performs best in terms of both the solution quality and the execution time, as shown in Tables 1, 2, and 3. The periodic exchange scheme often completes its execution earlier than the non-interacting MMC scheme. However, as the problem size grows and the number of subgraphs (subnetworks) increases, it is likely to be trapped at a local minimum (Table 1 for $N = R = 8$ and Table 3 for $N = R = 16$). It can be observed that the MMC PSA performs much better in terms of the solution quality and/or the execution time than the SMC PSA in most cases. This superiority of the MMC PSA becomes more visible as the number of PEs employed, N, increases. The speed-ups reported in the tables

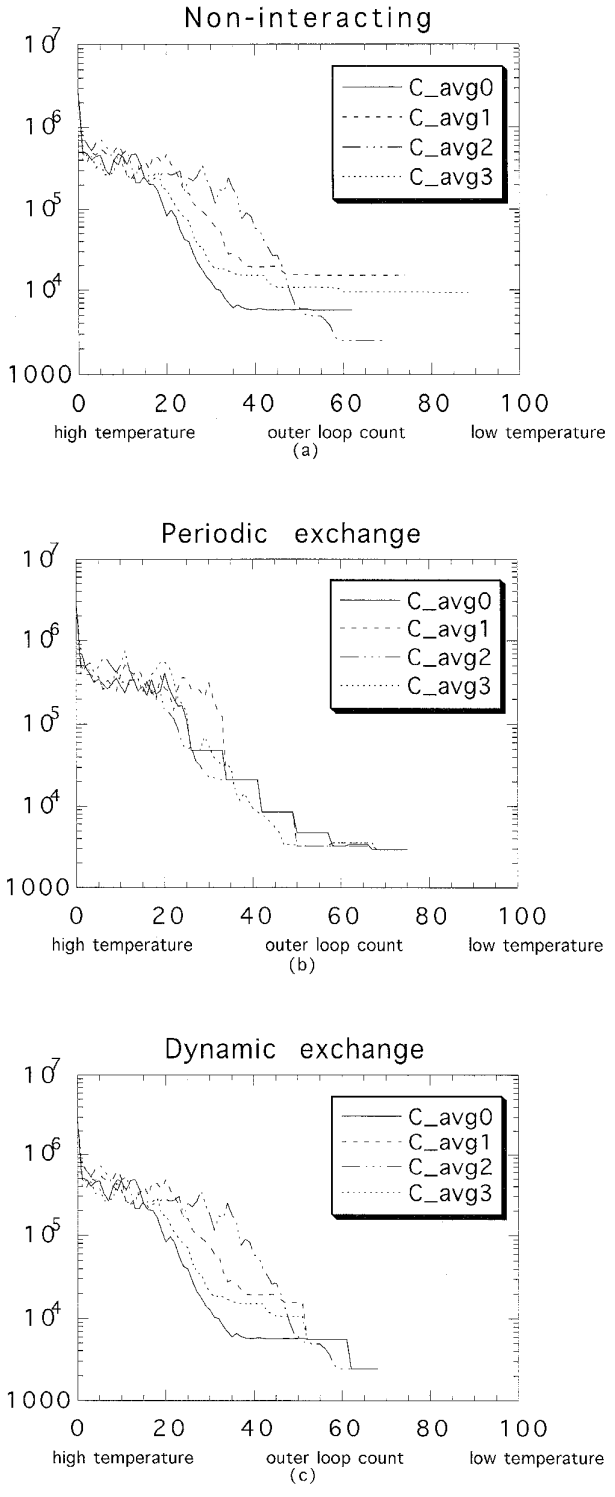


Fig. 6. Annealing curves of MMC PSAs ($N = 4$ for G118) (a) noninteracting, (b) periodic exchange, (c) dynamic exchange.

are based on different qualities of solutions due to the random nature of SA. However, note that the solution quality by the MMC PSAs is better than that by the SMC PSAs in almost all cases (all cases for the dynamic exchange scheme). Therefore, it is expected that higher speed-ups would be achieved if we somehow force the same quality of solution in both of MMC and SMC PSAs.

These observations may be explained as follows. First, since multiple search paths (chains) are followed in the MMC PSA, the probability of searching the local space where a global (or near) optimal solution is located would be higher for the MMC PSAs. Once most PEs (at least one PE) get on the right track, the annealing toward the near global optimum solution could be accelerated especially in the interacting MMC PSAs, i.e., a shorter execution time. Second, the SMC PSA requires more frequent communications (broadcast, collect, and exchange) among PEs since each PE needs to get the updated configuration and cost for every perturbation. This less communication overhead of the MMC PSA is another major factor which contributes to their better performance. Since the communication time is proportional to the number of PEs, N, the SMC PSA performs significantly worse than the MMC PSA, especially for a larger N. Third, in the MMC PSA, the P/E/D steps are fully (equivalently speaking) parallelized while in the SMC PSA the perturbation and decision (with updating the new graph partition in sequence) may not be executed fully in parallel.

The typical annealing curves (average cost vs. temperature) for the MMC PSAs are shown in Fig. 6. Each curve in the non-interacting scheme represents a normal sequential annealing curve, but only one of them leads to the final (best) solution. Note that the PE which completes annealing last has not found the best solution. The periodic exchange MMC PSA may reduce the redundant or unnecessary computation. However, the periodic exchange does not always result in positive effects as can be seen in Tables 1 and 3 and the annealing curve in Fig. 6b. Overly frequent information exchanges can lead to oscillation, hampering careful annealing processes and eventually deteriorating the convergence. In other words, only the sufficiently annealed solutions are worthwhile to be exchanged. This is why the dynamic exchange scheme improves the solution quality and also the execution time in most cases (Tables 1, 2 and 3).

Fig. 7 shows how the solution quality varies depending on the reduction factor, $g(N)$, for the G30 and G118. As the number of PEs (or Markov chains), N, increases, the solution quality (average cost) is improved to a certain point and then is degraded in general. A more drastic reduction factor like $1/N$ usually has the turning point at small N (compared to $\frac{1}{\sqrt{N}}$ or $\frac{1}{\log_2 N+1}$). That is, there can be a trade-off between the solution quality and execution time (remember that the speed-up, S, is almost inversely proportional to $g(N)$).

The problem dependencies of the SMC and MMC PSAs are compared in Fig. 8. In these experiments, the system parameters (except $t = t_p + t_e$) like communication time, obtained from the iPSC/2, were used. Fig. 8a shows that for a certain value of t, it would be possible for the speed-up of the SMC PSA to decrease after some point as N increases. Also, the derivatives of S_{mmc} and S_{smc} with respect to t (refer to (8) and (9)) are plotted in Fig. 8b. It is obvious that S_{mmc} is much less problem-dependent than S_{smc}.

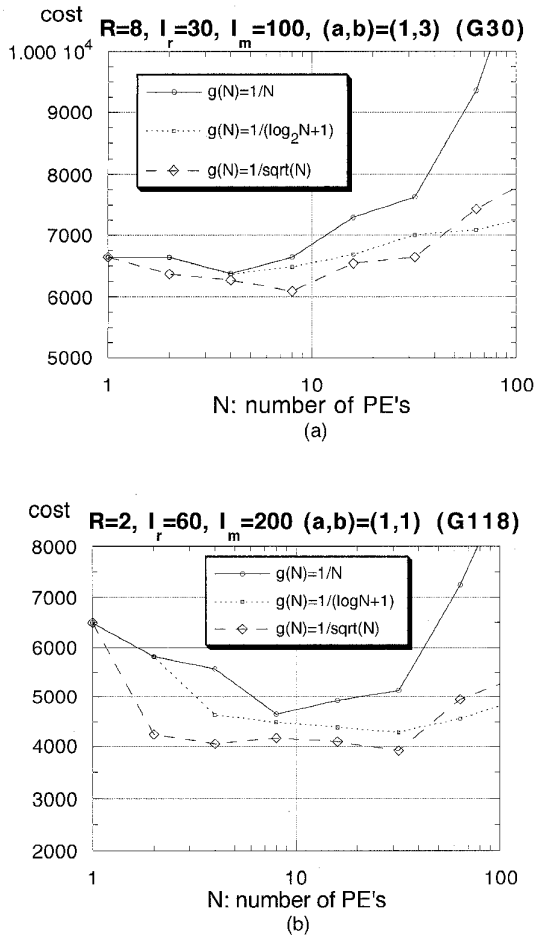


Fig. 7. Effect of reduction factor $g(N)$ for MMC PSA (dynamic exchange) for (a) G30, (b) G118 ($N > 32$: sequential simulation results).

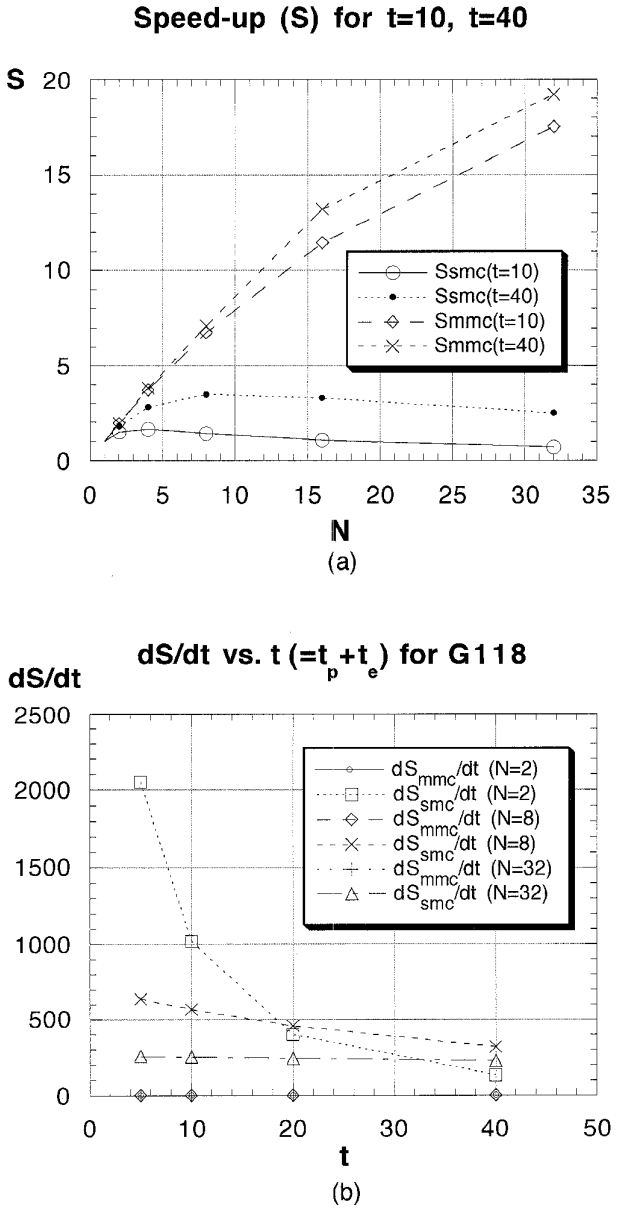


Fig. 8. (a) Speed-up (S) and (b) $\partial S/\partial t$ for SMC and MMC PSA. Note that the three curves for MMC in (b) are overlapped at the bottom (SMC: single move scheme, MMC: dynamic exchange scheme).

4.4.4 Asynchronous MMC PSA

The improvement factor, r, reported in Tables 4, 5, and 6 should be considered to be *conservative* since the solutions by the asynchronous MMC PSA are mostly better than those by its counterpart, the synchronous MMC PSA.

It is clear that the asynchronous MMC PSA outperforms the synchronous MMC PSA in speed as analyzed in Section 3.2. For the cases considered in this study, performance of MMC PSAs has been improved up to 43% by the asynchronous scheme in execution speed (refer to Table 4b). As can be seen in Table 4, r becomes larger as the global state access frequency, M, increases. Also, a larger problem (e.g., a larger graph, a larger number of partitions, a more complicated cost function) tends to result in a larger improvement factor since the segment length varies more relative to its mean when the problem (graph) size increases.

From Table 5, it can be observed that r increases as the number of Markov chains or PEs increases. This is because the maximum value (outcome) of segment length would be larger in general when there is a larger number of segments. This improvement in execution speed is also partially due to the communication overhead which grows with N faster in the synchronous MMC PSA than in the asynchronous MMC PSA (refer to Tables 6b and 6d). For reference, the speed-up, S, of the asynchronous MMC PSA over the sequential SA is included in Table 5. It is seen that, in most cases, a high parallelization efficiency is obtained.

The *percentage* idle and communication (global state access) times (i.e., normalized by the total execution time) with N or M varied are provided in Table 6. For a fair comparison, the (absolute) idle and communication times are normalized by the total execution time which changes with N and M. First, it is clear that the percentage idle time outweighs the percentage communication time in both

TABLE 4
EFFECT OF GLOBAL STATE ACCESS FREQUENCY (M)

M	Synchronous		Asynchronous		r
	cost	t_{syn} [msec]	cost	t_{asyn} [msec]	
6	538	4845	530	4823	1.00
12	557	4971	514	4538	1.09
25	557	5224	528	4929	1.06
50	638	5248	565	4652	1.13
125	622	5909	581	4850	1.22

(a) G30 (N = 8, R = 4, g(N) = 1/ N, (x, y)=(2, 3))

M	Synchronous		Asynchronous		r
	cost	t_{syn} [msec]	cost	t_{asyn} [msec]	
6	74502	49015	64215	47780	1.03
12	67531	48087	67641	45788	1.05
25	74284	47360	74092	43645	1.09
50	73156	51944	72426	42596	1.21
125	83968	59831	76056	41923	1.43

(b) G118 (N = 16, R = 16, g(N) = 1/ N, (x, y)=(2, 3))

M	Synchronous		Asynchronous		r
	cost	t_{syn} [msec]	cost	t_{asyn} [msec]	
6	5.03e6	164983	4.77e6	155794	1.05
12	5.64e6	165943	4.99e6	154472	1.07
25	5.72e6	172550	4.47e6	158302	1.12
50	5.76e6	194767	5.69e6	159645	1.22
125	5.70e6	221558	4.96e6	157581	1.41

(c) G200 (N = 16, R = 8, g(N)= 1/ N, (x, y)=(2, 3))

TABLE 5
EFFECT OF THE NUMBER OF MARKOV CHAINS OR PEs (N)

N	Synchronous		Asynchronous		r	S
	cost	t_{syn} [msec]	cost	t_{asyn} [msec]		
2	560	18845	555	18681	1.01	2.3
4	624	9813	587	9650	1.02	4.5
8	638	5248	565	4652	1.13	9.3
16	587	3165	584	2662	1.19	16.3

(a) G30 (M = 50, R = 4, g(N)= 1/ N, (x, y)=(2, 3))
(cost = 644, time = 43462 when N = 1)

N	Synchronous		Asynchronous		r	S
	cost	t_{syn} [msec]	cost	t_{asyn} [msec]		
2	63212	542750	58615	533103	1.02	2.0
4	67860	226031	59646	218334	1.04	4.9
8	74257	106792	74313	97322	1.10	11.0
16	73156	51944	72426	42596	1.21	25.1

(b) G118 (M = 50, R = 16, g(N)= 1/ N, (x, y)=(2, 3))
(cost = 64256, time = 1069613 when N = 1)

N	Synchronous		Asynchronous		r	S
	cost	t_{syn} [msec]	cost	t_{asyn} [msec]		
2	4.82e6	1034782	4.33e6	1019995	1.01	2.1
4	5.14e6	524878	4.31e6	495167	1.06	4.3
8	5.78e6	282728	5.43e6	254350	1.11	8.4
16	6.13e6	194767	5.93e6	159645	1.22	13.3

(c) G200 (M = 50, R = 8, g(N)= 1/ N, (x, y)=(2, 3))
(cost = 4.77e6, time = 2131867 when N = 1)

PSAs, especially for larger problem sizes. Second, both the percentage idle and communication times are much larger in the synchronous MMC PSA than in the asynchronous MMC PSA (refer to the ratios in the tables). Therefore, the major factor which distinguishes the asynchronous and synchronous MMC PSAs is the idle time (according to these two observations). Third, it can be seen that the percentage idle time increases as the global state access frequency, M, or the number of PEs (Markov chains), N, increases, especially in the synchronous scheme. Fourth, the ratio of the percentage idle times, *synchronous* over *asynchronous*, increases significantly with the global state access frequency, M, and the number of PEs, N. The idle time increases linearly with M in the synchronous scheme while it is proportional to $\sqrt{M}$ in the asynchronous scheme. The idle time in the asynchronous MMC PSA, which occurs only after the last iteration, is less sensitive to N than the (accumulated) idle time in the synchronous MMC PSA. Fifth, the ratio of the communication times, synchronous over asynchronous, increases with N but decreases with M. The communication time increases with both of M and N in both schemes. However, in the asynchronous scheme, it increases faster with M than with N. That is, the communication time in the asynchronous scheme is proportional to M (the number of global accesses), but not to N because the simultaneous access to the global state (in this case the communication time would also depend on N) rarely occurs during the execution of SA.

It is also observed that on average better solutions are found by the asynchronous MMC PSA. This is mainly because PEs in the asynchronous scheme do not start each segment from the same state (solution) and, therefore, they have a better chance *not* to miss the global optimal solution.

Tables 4, 5, and 6 are obtained by synchronous and asynchronous implementations of the periodic exchange MMC PSAs for a better control of M. The dynamic exchange MMC PSA has also been implemented asynchronously and its performance has been compared with that of its synchronous counterpart. In general, similar behaviors were observed. The asynchronous version of the dynamic exchange MMC PSA achieves improvement (r) of up to 24% (with equivalent qualities of solutions) over the synchronous version for the cases considered.

5 CONCLUSION

Although SA can find a global (near) optimal solution, its use has been severely limited in practice mainly due to the long computation time requirement. In order to shorten the computation time, parallelization of SA has been attempted in various applications. However, these efforts have not been very successful since they tried to preserve the single Markov chain in most cases (SMC PSA) and, therefore, the parallelism exploitable was limited. Although the idea of following multiple Markov chains (MMC PSA) was sketched in the literature, it was not fully developed and had a significant drawback (too frequent communication).

In this study, we have developed new MMC PSAs, especially the dynamic exchange scheme. Both synchronous and

TABLE 6
PERCENTAGE COMMUNICATION AND IDLE TIMES OF MMC PSAs: $(g(N)=\frac{1}{N}, (t_{i_s}, t_{i_a}), (t_{c_s}, t_{c_a})$: PERCENTAGES OF IDLE AND COMMUNICATION TIMES

M	Synchronous		Asynchronous		t_i ratio	t_c ratio
	t_{i_s} (%)	t_{c_s}(%)	t_{i_a}(%)	t_{c_a}(%)		
6	9.7	0.37	7.2	0.04	1.3	9.0
12	8.0	0.68	4.7	0.13	1.7	5.2
50	14.2	2.62	6.3	0.54	2.3	4.9
125	23.3	5.94	6.1	1.38	3.8	4.3

(a) G30 (N = 8, R = 4)

N	Synchronous		Asynchronous		t_i ratio	t_c ratio
	t_{is} (%)	t_{cs}(%)	t_{ia}(%)	t_{ca}(%)		
2	3.7	0.24	1.7	0.11	2.1	2.1
4	7.6	0.74	3.3	0.25	2.3	3.0
8	14.2	2.62	6.3	0.54	2.3	4.9
16	32.0	7.95	8.2	1.09	3.9	7.3

(b) G30 (M= 50, R = 4)

M	Synchronous		Asynchronous		t_i ratio	t_c ratio
	t_{i_s} (%)	t_{c_s}(%)	t_{i_a}(%)	t_{c_a}(%)		
6	10.9	0.37	4.9	0.004	2.2	9.7
12	12.5	0.69	4.6	0.07	2.7	9.6
50	23.3	2.54	4.9	0.26	4.7	9.6
125	35.4	5.63	5.8	0.65	6.1	8.6

(c) G118 (N = 16, R = 16)

N	Synchronous		Asynchronous		t_i ratio	t_c ratio
	t_{is} (%)	t_{cs}(%)	t_{ia}(%)	t_{ca}(%)		
2	7.9	0.04	2.7	0.02	2.9	2.4
4	13.5	0.15	3.4	0.04	3.9	3.6
8	18.5	0.58	3.5	0.10	5.2	5.7
16	23.3	2.54	4.9	0.26	4.7	9.6

(d) G118 (M = 50, R = 16)

asynchronous implementations have been considered. Their performance has been analyzed and experimentally compared to other PSAs for graph partitioning. The following conclusions may be drawn based on our extensive experimental results:

- The MMC PSAs can find a solution of better or equivalent quality faster than the SMC PSAs.
- The performance of the MMC PSAs is much less problem-dependent than that of the SMC PSAs.
- Among the MMC PSAs, the dynamic exchange scheme outperforms others in speed and solution quality.
- It is possible to reduce the execution time of the MMC PSA significantly by employing asynchronous information exchange.
- The quality of solution by the asynchronous MMC PSA is as good as (or even better than) that by the synchronous version.
- The speed-up of the asynchronous MMC PSA over the synchronous version becomes larger for a greater number of PEs (or Markov chains) involved or a higher global state access frequency.

REFERENCES

[1] S. Kirkpatrick, C.D. Gelatt Jr., and M.P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, pp. 671-680, 1983.

[2] U. Faigle and R. Schrader, "On the Convergence of Stationary Distributions in Simulated Annealing Algorithms," *Information Processing Letters*, vol. 27, pp. 189-194, 1988.

[3] A. Corana, M. Marchesi, C. Martini, and S. Ridella, "Minimizing Multimodal Functions of Continuous Variables with the Simulated Annealing Algorithm," *ACM Trans. Mathematical Software*, vol. 13, no. 3, pp. 262-280, Sept. 1987.

[4] F. Romeo and Sangiovanni-Vincentelli, "Probabilistic Hill Climbing Algorithms: Properties and Applications," *Proc. Chapel Hill Conf. VLSI*, Chapel Hill, pp. 393-417, 1985.

[5] R.H.J.M. Otten and L.P.P.P. van Ginneken, "Floorplan Design Using Simulated Annealing," *Proc. IEEE Int'l Conf. Computer-Aided Design*, Santa Clara, Calif., pp. 96-98, 1984.

[6] B. Hajek, "Cooling Schedules for Optimal Annealing," *Mathematics of Operations Research*, vol. 13, pp. 311-329, 1988.

[7] E. Aarts and J. Korst, *Simulated Annealing and Boltzman Machines*, pp. 95-128. John Wiley & Sons, 1989.

[8] S.A. Kravitz and R.A. Rutenbar, "Placement by Simulated Annealing on a Multi-Processor," *IEEE Trans. Computer-Aided Design*, vol. 6, pp. 534-549, July 1987.

[9] E.E. Witte, R.D. Chamberlain, and M.A. Franklin, "Parallel Simulated Annealing Using Speculative Computation," *IEEE Trans. Parallel and Distributed Systems*, vol. 2, pp. 83-494, Oct. 1991.

[10] P. Banerjee, M.H. Jones, and J.S. Sargent, "Parallel Simulated Annealing Algorithms for Cell Placement on Hypercube Multiprocessors," *IEEE Trans. Parallel and Distributed Systems*, vol. 1, no. 1, pp. 91-106, Jan. 1990.

[11] P. Roussel-Ragot and G. Dreyfus, "A Problem Independent Parallel Implementation of Simulated Annealing: Model and Experiments," *IEEE Trans. Computer-Aided Design*, vol. 9, no. 8, pp. 827-835, Aug. 1990.

[12] D.R. Greening, "Parallel Simulated Annealing Techniques," *Physica*, vol. D42, pp. 293-306, 1990.

[13] K.S. Natarajan "Graph-Partitioning on Shared-Memory Multiprocessor Systems," *Proc. Int'l Conf. Parallel Processing*, vol. 3, pp. 120-124, Aug. 1991.

[14] V.C. Barbosa and E. Gafni, "A Distributed Implementation of Simulated Annealing," *J. Parallel and Distributed Computing*, vol. 6, pp. 411-434, 1989.

[15] E.H. Aarts and J.H.M. Korst, "Boltzmann Machines as a Model for Parallel Annealing," *Algorithmica*, vol. 6, pp. 437-465, 1991.

[16] A. Casotto and A.L. Sangiovanni-Vincentelli, "Placement of Standard Cells Using Simulated Annealing on the Connection Machine," *Proc. IEEE Int'l Conf. Computer-Aided Design*, pp. 350-353, Santa Clara, Calif., 1987.

[17] K.-G. Lee and S.-Y. Lee, "Parallel Simulated Annealing for Finding Minima of Functions on a Hypercube Multiprocessor," *SIAM Ann. Meeting*, Chicago, July 1990.

[18] S.-Y. Lee, H.D. Chiang, K. -G. Lee, and B.Y. Ku, "Parallel Power System Transient Stability Analysis on Hypercube Multiprocessors," *IEEE Trans. Power Systems*, vol. 6, no. 3, pp. 1,337-1,343, Aug. 1991.

[19] M.D. Huang, F.Romeo, and A. Sangiovanni-Vincentelli, "An Efficient General Cooling Schedule for Simulated Annealing," *Proc. Int'l Conf. Computer-Aided Design*, pp. 381-384, Nov. 1986.

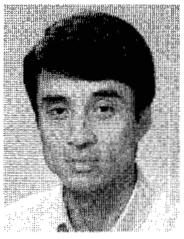
[20] D.E. Goldberg, *Genetic Algorithms*. Addison-Wesley, 1989.

[21] M.R. Irving and M.J.H. Sterling, "Optimal Network Tearing Using Simulated Annealing," *IEE Proc.*, vol. 137, no. 1, pp. 69-72, Jan. 1990.

[22] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco: W.H. Freeman, 1979.

[23] K.-G. Lee and S.-Y. Lee, "Efficient Parallelization of Simulated Annealing Using Multiple Markov Chains: An Application to Graph Partitioning," *Proc. Int'l Conf. Parallel Processing*, St. Charles, Ill., Aug. 1992.

[24] S.-Y. Lee and K.-G. Lee, "Asynchronous Communication of Multiple Markov Chains in Parallel Simulated Annealing," *Proc. Int'l Conf. Parallel Processing*, St. Charles, Ill., Aug.1992.



Soo-Young Lee received a BS degree in electronics engineering from Seoul National University in 1978, an MS degree in electrical and electronics engineering from the Korea Advanced Institute of Science in 1980, and a PhD degree in electrical and computer engineering from the University of Texas at Austin in 1987. From 1980 to 1983, he was an instructor in the Department of Electronics Engineering, Kyung-Pook National University, Taegu, Korea. From 1987 to 1994, he was on the faculty of the School of Electrical Engineering, Cornell University. In 1995, he joined the Department of Electrical Engineering, Auburn University, where he is an associate professor. Dr. Lee's research interests include parallel architectures and algorithms, task mapping, multiprocessor communication, performance analysis, image processing and its applications, proximity effect correction in E-beam lithography, etc. He is on the editorial board of the *ISCA International Journal of Computers and Their Applications*. He is a senior member of the IEEE and a member of Tau Beta Pi and was included in *Who's Who Among Asian Americans* in 1994.

Kyung Geun Lee received his BS degree in electronics engineering from Seoul National University and his MS degree in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST) in 1981 and 1983, respectively. In 1992, he received his PhD degree in electrical engineering from Cornell University, Ithaca, New York. He is a principal engineer in the Network Research Group at Samsung Electronics Co., Seoul, Korea, where he has been involved in developing asynchronous transfer mode (ATM) switches since 1993. His current research interests include the practical aspects of parallel processing, especially in broadband ISDN switching systems, and high-speed networking. Dr. Lee is a member of the IEEE and the KIEE (Korea Institute of Electronic Engineers).