Trading Accuracy for Speed in Parallel Simulated Annealing with Simultaneous Moves

M. D. Durand[*] Steve R. White[†]

May 7, 1999

Abstract

A common approach to parallelizing simulated annealing to generate several perturbations to the current solution simultaneously, requiring synchronization to guarantee correct evaluation of the cost function. The cost of this synchronization may be reduced by allowing inaccuracies in the cost calculations. We provide a framework for understanding the theoretical implications of this approach based on a model of processor interaction under reduced synchronization that demonstrates how errors in cost calculations occur and how to estimate them. We show how bounds on error in the cost calculations in a simulated annealing algorithm can be translated into worst-case bounds on perturbations in the parameters which describe the behavior of the algorithm.

Keywords: Parallel simulated annealing; Metropolis algorithm; Boltzmann distribution.

Simulated annealing (SA) is an iterative method for finding approximate solutions to intractable combinatorial optimization problems. For parallel implementations of SA based on simultaneous moves, synchronization is a bottleneck because it reduces the amount of available parallelism and increases the overhead. Synchronization costs can be reduced by allowing processors to use out-of-date state information to evaluate the quality of intermediate solutions at the expense of introducing errors into the evaluation function. This approach assumes that SA will still generate a good solution if the size of the errors is controlled.

Since 1986, parallel implementations of SA using simultaneous moves have been reported in the literature, using many different heuristics to control the size of the error. However, to our knowledge, very little has been published that analyzes the nature and impact of these errors theoretically. In this paper, we model the error that occurs on multiprocessors and analyze this error using techniques from statistical physics. First, a formal model of parallel simulated annealing (PSA) based on simultaneous moves with reduced synchronization is introduced. We describe how error in the change in energy occurs in this model and discuss implementation issues that affect its size. The effect of such error on the behavior of SA can be understood through its impact on the frequency with which the algorithm visits states in the solution space. We prove a theorem that

[*]Contact author: Department of Molecular Biology, Princeton University, Princeton, NJ 08544, USA, *durand@cs.princeton.edu*

[†]IBM T. J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598, USA, *srwhite@watson.ibm.com*

relates bounds on errors in the change in energy to bounds on the error in the associated probability distribution of states of a system at equilibrium. These bounds on the probability distribution allow us to derive bounds on macroscopic quantities such as the average energy in the presence of error. Finally, we show that this theorem can be applied to any type of error in the change in energy, not just errors which result from asynchrony in parallelism. These results provide a framework for understanding the tradeoff between speed and accuracy in PSA implementations by modeling the nature and consequences of permitting cost function error in order to reduce synchronization costs.

The rest of the paper proceeds as follows. In Section 1, serial simulated annealing is described in more detail. Parallel simulated annealing is introduced in Section 2 and a brief survey of related work is presented. In Section 3, a model of processor interaction in PSA is introduced. The roles of synchronization and error are discussed in the context of this model. Results are presented that relate the behavior of ordinary SA and PSA when errors in the change in energy are allowed. We discuss the limitations of this and ideas for future work.

1 Serial Simulated Annealing

Simulated annealing [5, 20] is a heuristic for solving prohibitively large instances of discrete combinatorial optimization problems. Let the state space, $S = \{s\}$, be the set of all possible solutions of a given combinatorial optimization problem. The move set, $\{m\}$, is the set of all operations that generate one solution from another by moving a component of the solution. The probability of generating state s from state t is q_{st}. If state t can be generated from state s in one move, then $q_{st} > 0$, otherwise $q_{st} = 0$. We require that the state space be connected; that is, every state can be reached from every other state in a finite number of moves. Let the distance $\lambda(s,t)$, between two states, s and t, be the minimum number of transitions needed to reach t from s (or vice versa). Then the *diameter*, $d = \max_{s,t \in S}\{\lambda(s,t)\}$, of a state space, S, is the maximum distance between any pair of states, taken over all possible pairs of states in S. To determine the optimal solution to our problem, we define a function, $E_s = E(s)$, that specifies the cost of every state. The optimal solutions are then those states which minimize E.

We illustrate these ideas using a specific problem, graph partitioning. Although there are specialized algorithms that solve this problem more efficiently than SA, we use graph partitioning as an example because it is extremely simple to describe and reason about and yet has many of the properties of VLSI layout, a real world problem which SA is frequently used to solve. In graph partitioning, the object is to separate the N vertices of an undirected graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, into two subsets of equal size, $\mathcal{V}_1$ and $\mathcal{V}_2$, with a minimum number of edges crossing the boundary between the two subsets (the cutset.) S is the set of all possible assignments of vertices to $\mathcal{V}_1$ or $\mathcal{V}_2$. A new state is obtained from the current by reassigning a vertex from its current subset to the other. Thus, $q_{st} > 0$ if states s and t differ by the assignment of one vertex. Clearly, the state space is connected since any state can be obtained from any other state by a series of vertex moves. The most distant pairs of states are those where every vertex is assigned to a different subset in the two states, requiring N vertex moves to get from one state to the other. Thus, the diameter of this space is N. In order to keep the partition from becoming too unbalanced, the cost is defined to be the sum of the cutset size and a term which discourages unbalanced partitions:

$$E = |\{(w_1, w_2)|w_1 \in \mathcal{V}_1 \wedge w_2 \in \mathcal{V}_2 \wedge (w_1, w_2) \in \mathcal{E}\}| + \alpha \cdot (|\mathcal{V}_1| - |\mathcal{V}_2|)^2 , \tag{1}$$

where α is a constant which controls the relative importance of the cutset and the partition imbalance. If a heuristic used to solve this problem yields an unbalanced partition as its final solution, a greedy algorithm can be used to rebalance the subsets.

Notice that it is not generally true that every move may be applied to every state. The domain a move, the set of states on which it is defined, may be a proper subset of S. In the move set for graph partitioning described above, for example, every move may be applied to every state. An alternate move set for this problem is to select a vertex from each subset and exchange them. In this case, the domain of the move "swap w_1 and w_2" is the set of states such that $w_1 \in \mathcal{V}_1 \wedge w_2 \in \mathcal{V}_2$, or vice versa. This move cannot be applied to states that have w_1 and w_2 in the same subset.

Iterative heuristics seek an optimal solution to a combinatorial optimization problem by starting with an initial solution, often chosen at random, and improving it by making stepwise changes. If only changes that decrease the cost ("downhill moves") are allowed, the iterative method will get stuck in local minima. Simulated annealing exploits a physical analogy to escape these local minima by making moves that increase the cost ("uphill moves") in a controlled way.

Physical annealing is the process of slowly cooling an ensemble of molecules to obtain the most ordered or lowest energy arrangement of the molecules (typically, a crystal). When the ensemble is at equilibrium at a fixed temperature, the probability, $P(s)$, that the system will be in state s with energy E_s is described by the *Boltzmann distribution*,

$$P(s) = \frac{e^{-E_s/T}}{\sum_t e^{-E_t/T}} , \tag{2}$$

where T is a parameter proportional to the temperature [23]. "Slow cooling" means lowering the temperature slowly enough so that the system always stays close to this equilibrium distribution. In the analogy, the combinatorial optimization problem corresponds to a system of molecules and the cost function to the energy of the system. The optimal solution (the "crystal") is obtained by simulating slow cooling; that is, by sampling the state space according to the Boltzmann distribution. A parameter analogous to the temperature, T, is decreased as the algorithm progresses. Simulated annealing approximates the Boltzmann distribution using the *Metropolis* algorithm to sample states in S. Metropolis *et al.* [22] proved that the Boltzmann distribution could be simulated by repeatedly generating a new state, t, from the current state, s, and including the new state in the sample with probability $P_{st} = \min(1, e^{-\Delta E_{st}/T})$, where $\Delta E = E_t - E_s$. In other words, the Metropolis algorithm accepts all downhill moves. Uphill moves are accepted with probability $e^{-\Delta E_{st}/T}$, so that the increases in energy that may be accepted decrease as T decreases. The expected cost of solutions sampled by the algorithm decreases also. If T is decreased sufficiently slowly, the algorithm will converge convergence in probability to the globally minimum energy states [25]. Final solution quality depends on the *annealing schedule*, the initial temperature and the rate at which the temperature is reduced. Thus, the behavior and convergence of simulated annealing is characterized by the probability distribution of sampled states at each stage of the algorithm. Theoretical analyses of simulated annealing, whether based on statistical physics and on Markov chains, rely on this observation [25].

2 Parallel Simulated Annealing with Simultaneous Moves

In order to improve SA performance, many parallel versions of SA have been developed(see [1, 7, 13] for surveys). An approach that is application independent and allows the exploitation of a reasonable amount of parallelism is to generate and evaluate several moves simultaneously [2, 3, 4, 6, 14, 18, 19, 21, 24, 26]. Since computing ΔE requires global state information, processors manipulating the state simultaneously may interfere with each other. For example, in the graph partitioning problem, when a vertex is moved from one subset to the other, the resulting change in the cutset will depend on which side of the partition all of the adjacent vertices are currently located. If one of those neighboring vertices is reassigned, the energy calculation will be affected.

Interactions between processors can be prevented by guaranteeing that moves that are scheduled simultaneously do not share any state information, either by locking all components of the state that contribute to the calculation of the change in energy or by partitioning the state into independent subsets. Unfortunately, many approaches that prevent processor interaction were characterized by poor performance because of high synchronization overhead and a limited amount of available parallelism [6, 8, 21].

An alternative is to allow processors to compute ΔE using out-of-date information but to limit the size of the error that can occur as a result. When each processor manipulates a different part of a single, global state description, errors in the change in energy are due only to the simultaneous actions of other processors. This error is *temporary* because the state description does not accumulate out-of-date information. Experimental results show that temporary error has little impact on final solution quality [3, 4, 26]. On distributed memory architectures, the state description is typically replicated. The state is partitioned and each processor is constrained to attempt moves within its own subset. However, every processor has a description of the entire state in order to calculate ΔE. The replicated copies become increasingly out-of-date as the processors propose and accept moves and must be periodically updated.

This *cumulative* error has a noticeable impact on final solution quality. The experimental literature (see [7] for a survey) suggests that cumulative error increases with the number of processors and the number of accepted moves between updates and is sensitive to the network architecture and the communication protocol used. A number of experimental studies explored the tradeoff between communication costs and final solution quality, seeking the minimum synchronization required to obtain acceptable results. Given this observed relationship between reduced synchronization and solution quality, exactly how is error occurring and how does it affect algorithm behavior?

Some early attempts to answer this question [2, 3, 17, 19] have estimated average accumulated error over a series of parallel moves experimentally by comparing the actual change in energy, the change in global energy between two updates, with the perceived change in energy, the sum of the values of ΔE associated with all accepted moves between those updates. Such estimates of the accumulated error have been used to select the number of moves between updates adaptively [2, 17]. However, because this approach does not focus on error in individual accept/reject decisions, it can not be used to study the impact of error in ΔE on transition probabilities. Yet, it is the change in transition probabilities that perturbs the probability distribution of states and hence, changes the convergence properties of the algorithm.

Gelfand and Mitter [11] present a Markov approach that does relate error in ΔE to convergence. However, they assume that the errors have a Gaussian distribution. Since this assumption is not supported by a model relating processor interaction to properties of the error in ΔE, it is not clear under what conditions their analysis can be used to model actual PSA programs or to examine the tradeoff between speed and accuracy.

Grover [15, 16] presents a serial annealing algorithm for standard cell placement where error occurs because ΔE is computed using a faster, approximate method. In an analysis that influenced our approach, Grover uses a statistical mechanical model to study the impact of errors in E on equilibrium properties of the system but does not relate errors in ΔE to algorithmic behavior. PSA algorithms are usually influenced by errors in ΔE, not E, so his results are not directly applicable to our problem.

3 Impact of Error

Previous studies of error in PSA with simultaneous moves have either failed to relate errors in the change of energy to a model of parallel execution or have not shown how error affects the acceptance of individual moves and, hence, the equilibrium distribution of states at fixed temperature. In the current paper, we present a model of processor interaction and use it to demonstrate how errors arise in individual moves. We then relate these errors to perturbations in the equilibrium behavior of the algorithm.

In [8, 10], we discussed the minimum synchronization required in a PSA algorithm based on simultaneous moves. A set of formal criteria for specifying which moves may be scheduled simultaneously under these conditions was presented. A move scheduling strategy that conforms to our criteria guarantees that the state description will never become corrupted. A PSA algorithm that uses such a strategy operates on the same set of states as SA and the energies of the states are not perturbed by error. Errors in calculating ΔE will occur, however, although these errors can be reduced through additional synchronization constraints.

In the current paper, we show how error in ΔE occurs in PSA algorithms adhering to our minimum synchronization criteria. The upper bound on this error depends on the application and the parallel architecture. We give an example of how the worst case error might be estimated using graph partitioning. Given an upper bound on the error in ΔE, proofs of worst-case bounds on the transition probability and the distribution of states are given. Since the behavior of SA depends on simulating equilibrium at all temperatures, the worst-case bounds on the probability distribution show that the effect of error on the behavior of perturbed SA will also be bounded. Finally, bounds on macroscopic properties such as the average energy at a given temperature are derived from the bounds on the probability distribution of states.

All previous PSA work surveyed here, has tacitly assumed that PSA, like SA, can be modeled as an irreducible, aperiodic Markov chain with a stationary equilibrium probability distribution at fixed temperature. In the following analysis, we make the same assumptions. We also assume that PSA algorithms schedule simultaneous moves that conform to our minimum synchronization criteria.

3.1 Errors Caused By Simultaneous Moves

As a result of executing several moves simultaneously, the state may change between the time a move is proposed and the time the move is accepted. For example, Figure 1 shows a processor

$$\begin{tikzcd}
 & \mathtt{t} & & & & & & \mathtt{v} \\
\mathtt{s} \arrow[ur, "m"] \arrow[r] & {} \arrow[r] & {} \arrow[r] & {} \arrow[r] & {} \arrow[r] & {} \arrow[r] & \mathtt{u} \arrow[ur, "m"] & \\
 & & & m_1, \ldots, m_\sigma & & & &
\end{tikzcd}$$

Figure 1: State Changes Due to Lack of Synchronization

considering move m from state s to state t. Between the beginning of the move and the accept/reject decision, moves $m_1, \ldots, m_\sigma$ have been accepted by other processors, changing the state to u. As a result the processor will accept or reject the move from u to v based on the difference in energy between s and t[1]. This leads to an error in the change in energy:

$$\Delta E'_{uv} = \Delta E_{st} = \Delta E_{uv} + \delta^m_{su} ,$$

where, $\Delta E'_{uv}$ is the erroneous change in energy between u and v. The processor is actually causing a change ΔE_{uv}, which differs from the *perceived* change in energy $\Delta E'_{uv} = \Delta E_{st}$ by an error

$$\delta^m_{su} = \Delta E_{st} - \Delta E_{uv} . \tag{3}$$

The size of the largest δ^m_{su} seen in a particular algorithm depends on the distance, σ, between s and u. The largest possible error in ΔE is

$$\delta_m = \max\{|\delta^m_{su}|\} ,$$

where the maximum is taken over all triples m, s, u such that s and u are in the domain of m and u is at most σ steps away from s. For the types of application problems simulated annealing is typically used to solve, the greater the value of σ, the greater the value of δ_m. First of all, as σ increases, so does the set of triples over which the maximum is taken, increasing the chances that a large one will be seen. In addition, in "real world" problems, the greater the distance between the true state and the perceived state, the greater the number of components required to compute E which have been displaced. The distance, σ, between s and u depends on the acceptance rate and the data distribution and update strategies. In general, error will occur more often at high temperature where the acceptance rate is higher.

[1]In [8, 10], we showed that if m_1 and m_2 may be scheduled simultaneously according to our minimum synchronization criteria and s is in the domain of m_1 then $t = m_2(s)$ is also in the domain of m_1. Therefor, if s is in the domain of m, u is also in the domain of m. The activities of other processors will not change the state to one that is not in the domain of m.

3.2 Estimating worst case error

Since error is application and implementation dependent, a specific upper bound on the error, δ_m, must be estimated for each SA implementation. In general, the size of δ_m will depend on the number of processors, the acceptance rate and the update frequency. As an example, we estimate δ_m for the graph partitioning problem. A processor is evaluating a move, m, that reassigns vertex w, changing the current state, s, to $t = m(s)$. Other processors are simultaneously considering the changing the state by reassigning other vertices. The size of the error that results when calculating ΔE will depend on n, the net number of neighbors of w that change sides. Given the cost function defined in Equation 1, the change in energy can be shown to be

$$\Delta E = (d_I - d_O) + 4\alpha(|\mathcal{V}_O| - |\mathcal{V}_I| + 1)\,, \tag{4}$$

where $\mathcal{V}_I$ is the current subset of w, $\mathcal{V}_O$ is the new subset of w if the move is accepted, d_I is the number of neighbors of w in $\mathcal{V}_I$ when the processor begins the move and d_O is the number of neighbors in $\mathcal{V}_O$. The perceived change in energy, ΔE_{st}, is given by Equation 4. The true change in energy is

$$\Delta E_{uv} = ((d_I - n) - (d_O + n)) + 4\alpha((|\mathcal{V}_O| + n) - (|\mathcal{V}_I| - n) + 1)\,,$$

where u is the state which results from the actions of other processors and $v = m(u)$. The error that occurs is $\delta^m_{su} = \Delta E_{st} - \Delta E_{uv} = 2n - 8\alpha n$.

The upper bound on this error depends on the properties of the graph and the properties of the implementation. The maximum error occurs when n is maximum. Clearly, n is never greater than the maximum degree of $\mathcal{G}$, $\mathcal{D}_m$. This upper bound can only be reached when w is the highest degree vertex, s is a state where all neighbors of w are in $\mathcal{V}_I$ and u is a state where all neighbors of w are in $\mathcal{V}_O$ (or vice versa). The upper bound on n is also limited by the maximum number of state changes that can occur during a move, σ_m. In the shared state model, $\sigma_m = (p-1)\cdot a$, where p is the number of processors and a is the number of moves that can be accepted by another processor during one move. In the replicated model, $\sigma_m = (p-1)\cdot a \cdot l$, where l is the number of attempted moves between updates. Combining the two limits on the maximum net number of vertices crossing the partition yields $n_m = \min(\sigma_m, \mathcal{D}_m)$. The minimum error in the cutset calculations in the graph partitioning problem is

$$\delta_m = |(2 - 8\cdot\alpha)\cdot n_{max}|\,.$$

3.3 The Effect of Error on Transition Probabilities

The error in the change in energy will also lead to an error in the transition probability, the probability of accepting a move from u to v:

$$P_{uv} = \min(1, e^{-\Delta E_{uv}/T})\,, \tag{5}$$

where $v = m(u)$. The transition probability in the presence of error, P'_{uv}, is the weighted average over all the possible errors which can occur. An error occurs whenever the system was in a state other than u when the processor begins its move. Every state, s, which is in the domain of m contributes to the error. For each of these states, s, let $\Pi_{um}(s)$ be the probability that a processor will consider changing the state from u to v using ΔE_{st}, where $v = m(u)$ and $t = m(s)$. $\Pi_{um}(s)$ depends on how easily the other processors can change the state from s to u. The perturbed transition probability is the summation of each erroneous transition probability, P_{st}, weighted by the probability that a transition was made from s to u:

$$P'_{uv} = \sum_{s} \Pi_{um}(s) \cdot P_{st}\,, \tag{6}$$

where $v = m(u)$ and $t = m(s)$.

Theorem 1 *The probability of making a transition from state u to state v at temperature, T, at equilibrium in the perturbed system is related to the unperturbed transition probability by:*

$$e^{-\delta_m/T} \cdot P_{uv} \;\leq\; P'_{uv} \;\leq\; e^{+\delta_m/T} \cdot P_{uv}\,.$$

Proof : An error in the change in energy, ΔE, does not necessarily lead to error in the decision to accept or reject the move. The effect of error on P_{uv} depends not only on the magnitude of the error but also on the signs of ΔE_{uv} and $\Delta E'_{uv}$. For example, if both ΔE_{uv} and $\Delta E'_{uv}$ are less than zero, then the correct action, accepting the move, will be taken even if the value of ΔE is incorrect.

Considering all the possible combinations of the signs for ΔE_{uv} and $\Delta E'_{uv}$, yields four different cases. Let s, t, u and v be four states as shown in Figure 1, where $t = m(s)$, $v = m(u)$ and u and s are in the domain of m. We will show that the erroneous transition probability, P_{st}, is bounded by

$$e^{-\delta_m/T} \cdot P_{uv} \;\leq\; P_{st} \;\leq\; e^{+\delta_m/T} \cdot P_{uv}\,. \tag{7}$$

for each of the four cases. Then by substituting these bounds into the right hand side of Equation 6, bounds on P'_{uv} are derived.

The *first case*, shown in Figure 2(a), occurs when $\Delta E_{uv} > 0$ and $\Delta E'_{uv} = \Delta E_{st} > 0$. In this case, both the actual move and the intended move are uphill moves. The error in the accept/reject decision comes from the difference in the accept probabilities $P_{uv} = e^{-\Delta E_{uv}/T}$ and $P'_{uv} = P_{st} = e^{-\Delta E_{st}/T}$. By Equation 3, we know that $\Delta E_{st} = \Delta E_{uv} + \delta^m_{su}$, so $P_{st} = e^{-\delta^m_{su}/T} \cdot P_{uv}$. Since δ_m is the maximum possible error, $e^{-\delta_m/T} \;\leq\; e^{-\delta^m_{su}/T} \;\leq\; e^{+\delta_m/T}$ and therefore $e^{-\delta_m/T} \cdot P_{uv} \;\leq\; P_{st} \;\leq\; e^{+\delta_m/T} \cdot P_{uv}$.

In the *second case* (Figure 2(b)), $\Delta E_{uv} > 0$ and $\Delta E'_{uv} = \Delta E_{st} \leq 0$. In this case, cost function error will cause the algorithm to always accept the move whereas the move should only be accepted with probability $e^{-\Delta E_{uv}/T}$. The perturbed transition probability differs from the true transition probability by $P_{st} = 1 = e^{+\Delta E_{uv}/T} \cdot e^{-\Delta E_{uv}/T} = e^{+\Delta E_{uv}/T} \cdot P_{uv}$. Since ΔE_{uv} and ΔE_{st} have different signs, it will always be true that $|\delta^m_{su}| \;\geq\; \Delta E_{uv}$. Since $\delta_m \;\geq\; |\delta^m_{su}|$, $-\delta_m \;\leq\; \Delta E_{uv} \;\leq\; +\delta_m$ and therefore Equation 7 holds when $\Delta E_{uv} > 0$ and $\Delta E_{st} < 0$. The *third case* ($\Delta E_{uv} \leq 0$, $\Delta E_{st} > 0$), shown in Figure 3(a), is analogous to the second case.

In the *fourth case*, shown in Figure 3(b), both ΔE_{st} and ΔE_{uv} represent downhill moves. Thus $P_{st} = P_{uv} = 1$ and there is no error in the transition probabilities. Therefore the bounds on P_{st} given in Equation 7 apply trivially here.

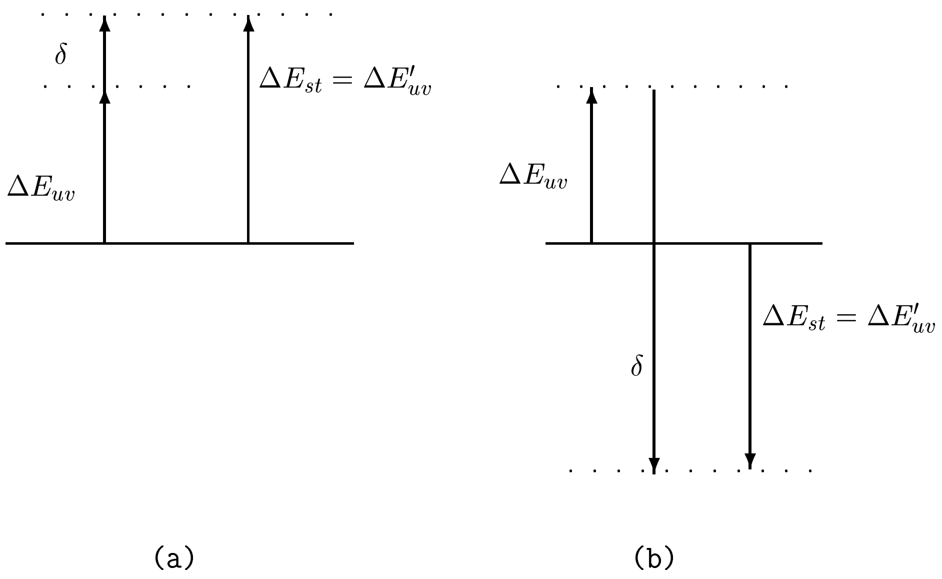
δ
$\Delta E_{st} = \Delta E'_{uv}$
ΔE_{uv}
ΔE_{uv}
$\Delta E_{st} = \Delta E'_{uv}$
δ
(a)
(b)

Figure 2: First and Second Cases ($\Delta E_{uv} > 0$)

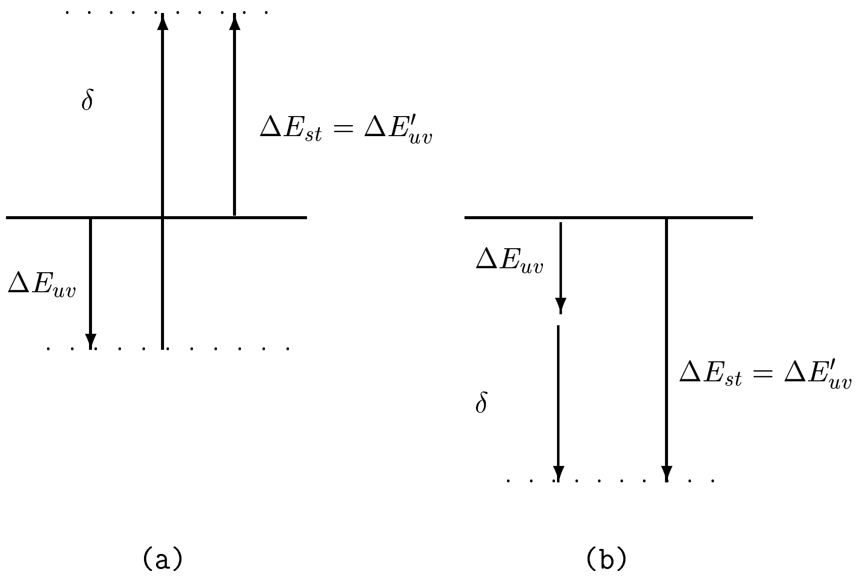
δ
$\Delta E_{st} = \Delta E'_{uv}$
ΔE_{uv}
ΔE_{uv}
$\Delta E_{st} = \Delta E'_{uv}$
δ
(a)
(b)

Figure 3: Third and Fourth Cases ($\Delta E_{uv} \leq 0$)

Now that the upper and lower bounds in Equation 7 have been proved for all four cases, those bounds can be substituted into Equation 6:

$$\sum_s \Pi_{um}(s) \cdot e^{-\delta_m/T} \cdot P_{uv} \;\le\; P'_{uv} \;\le\; \sum_s \Pi_{um}(s) \cdot e^{+\delta_m/T} \cdot P_{uv} \,.$$

Since $e^{-\delta_m/T}$ and $e^{+\delta_m/T}$ are independent of s and $\sum_s \Pi_{um}(s) = 1$, $e^{-\delta_m/T} \cdot P_{uv} \;\le\; P'_{uv} \;\le\; e^{+\delta_m/T} \cdot P_{uv}$ $\Box$

3.4 Bounds on the Probability of Occupying a State

The error in the transition probability will change the frequency with which states are visited. Using the bounds on the transition probability, P'_{st}, proved in Theorem 1, we show below that there are also bounds on $P'(s)$, the probability, in the presence of error, that the system is in state s.T he size of the impact that error in ΔE has on the final solution quality is related to how much the perturbed probability distribution, $P'(s)$, differs from the original probability distribution, $P(s)$.

Theorem 2 *Let SA be a simulated annealing algorithm and PSA be the associated perturbed annealing algorithm. Let δ_m be the maximum possible error. SA will have a state probability distribution, $P(s)$, in equilibrium at fixed temperature and PSA will have a probability distribution, $P'(s)$, in equilibrium at fixed temperature. Then $P'(s)$ will be related to $P(s)$ by:*

$$e^{-2d\delta_m/T} \cdot P(s) \;\le\; P'(s) \;\le\; e^{+2d\delta_m/T} \cdot P(s) \,, \tag{8}$$

where d is the diameter of the state space.

Proof : To prove this assertion we use an argument similar to that of Metropolis *et al.* [22], based on the same initial assumptions: the function for generating new states is symmetric ($q_{st} = q_{ts}$); any state can be reached from any other state by a finite sequence of moves and at equilibrium, $P(s) \cdot P_{st} = P(t) \cdot P_{ts}$, $\forall(s,t)$. Imagine an ensemble in which many identical problems are being solved simultaneously by the same error-prone SA algorithm. The probability of being in state s is then equivalent to the fraction of systems in the ensemble in state s. Let ν_s be the number of systems in s and P_{st} is the probability of accepting a move from s to t as defined in Equation 5. Similarly, ν'_s is the number of systems in state s' in using the error-prone algorithm and P'_{st} is the perturbed transition probability. Then, the probability of being in state s is:

$$P'_s = \frac{\nu'_s}{\sum_t \nu'_t} \,. \tag{9}$$

In an ensemble of systems at equilibrium, the number of systems going from s to t is equal to the number of systems going from t to s. Therefore, in an ensemble with error at equilibrium $\nu'_s \cdot q_{st} \cdot P'_{st} = \nu'_t \cdot q_{ts} \cdot P'_{ts}$, for any two adjacent states. Since $q_{st} = q_{ts}$,

$$\nu'_t = \frac{P'_{st}}{P'_{ts}} \nu'_s \,. \tag{10}$$

If s and t are not adjacent, then there exists a sequence of intermediate states such that a path exists from s to t, since the state space is connected. Let $x_1, \cdots, x_n$ be such a sequence of minimum length. In this case,

$$\nu'_t = \frac{P'_{x_n t}}{P'_{t x_n}} \cdots \frac{P'_{x_1 x_2}}{P'_{x_2 x_1}} \cdot \frac{P'_{s x_1}}{P'_{x_1 s}} \nu'_s \,. \tag{11}$$

Using the bounds on the transition probability given in Theorem 1, we can show that any fraction of the form $\frac{P'_{x_i x_{i+1}}}{P'_{x_{i+1} x_i}}$, where x_i and x_{i+1} are adjacent states, is bounded by

$$\begin{aligned} \frac{e^{-\delta_m/T}\, P_{x_i x_{i+1}}}{e^{+\delta_m/T}\, P_{x_{i+1} x_i}} \le \frac{P'_{x_i x_{i+1}}}{P'_{x_{i+1} x_i}} \le \frac{e^{+\delta_m/T}\, P_{x_i x_{i+1}}}{e^{-\delta_m/T}\, P_{x_{i+1} x_i}} \,, \\ e^{-2\delta_m/T}\, \frac{P_{x_i x_{i+1}}}{P_{x_{i+1} x_i}} \le \frac{P'_{x_i x_{i+1}}}{P'_{x_{i+1} x_i}} \le e^{+2\delta_m/T}\, \frac{P_{x_i x_{i+1}}}{P_{x_{i+1} x_i}} \,. \end{aligned} \tag{12}$$

By substituting the upper bound given in Equation 12 for each ratio on the right hand side of Equation 11, we get an upper bound on ν'_t in terms of P_{st} and P_{ts}:

$$\begin{aligned} \nu'_t &= \frac{P'_{x_n t}}{P'_{t x_n}} \cdots \frac{P'_{x_1 x_2}}{P'_{x_2 x_1}} \frac{P'_{s x_1}}{P'_{x_1 s}} \nu'_s \,, \\ &\le e^{+2\delta_m/T} \frac{P_{x_n t}}{P_{t x_n}} \cdots e^{+2\delta_m/T} \frac{P_{x_1 x_2}}{P_{x_2 x_1}} e^{+2\delta_m/T} \frac{P_{s x_1}}{P_{x_1 s}} \nu'_s \,, \\ &\le e^{+2(n+1)\delta_m/T} \frac{P_{x_n t}}{P_{t x_n}} \cdots \frac{P_{x_1 x_2}}{P_{x_2 x_1}} \frac{P_{s x_1}}{P_{x_1 s}} \nu'_s \,. \end{aligned} \tag{13}$$

Using tedious but straightforward algebra, it is easy to show that

$$\frac{P_{x_n t}}{P_{t x_n}} \cdots \frac{P_{x_1 x_2}}{P_{x_2 x_1}} \cdot \frac{P_{s x_1}}{P_{x_1 s}} = \frac{P_{st}}{P_{ts}} \,. \tag{14}$$

Notice that this is *not* the case for perturbed transition probabilities. The perturbed transition probabilities associated with intermediate states do not, in general, cancel out because the error associated with a transition from x_i to x_j is not necessarily the same as an error associated with a transition from x_j to x_i. This is why Equation 11 does not reduce to Equation 10. Substituting the right hand side of Equation 14 for the product in Equation 13 gives us an upper bound which is independent of intermediate states:

$$\nu'_t \le e^{+2(n+1)\delta_m/T} \frac{P_{st}}{P_{ts}} \nu'_s \,.$$

Furthermore, since $n+1$ is the minimum number of transitions required to get from s to t, $n+1$ is never greater than the diameter, d, of the space. Hence, we can generalize the upper bound to apply to any pair of states, s and t, in the space:

$$\nu_t \le e^{+2d\delta_m/T} \frac{P_{st}}{P_{ts}} \nu'_s \,.$$

A similar analysis gives a lower bound on ν'_t, yielding

$$e^{-2d\delta_m/T}\,\frac{P_{st}}{P_{ts}}\,\nu'_s \;\leq\; \nu'_t \;\leq\; e^{+2d\delta_m/T}\,\frac{P_{st}}{P_{ts}}\,\nu'_s\,. \tag{15}$$

Substituting the upper bound of Equation 15 for ν'_t in Equation 9, gives

$$P'_s \;\leq\; \frac{\nu'_s}{\sum_t e^{-2d\delta_m/T}\,\frac{P_{st}}{P_{ts}}\,\nu'_s} \;=\; e^{+2d\delta_m/T}\left(\sum_t \frac{P_{st}}{P_{ts}}\right)^{-1}\,. \tag{16}$$

Since $P_{st}=1$ if $E_s > E_t$ and $e^{-\Delta E_{st}/T}$ if $E_s \leq E_t$, we split the sum in Equation 16 into two parts; those states t for which $P_{st}=1$ and those states t for which $P_{ts}=1$. Thus,

$$\begin{aligned}\left(\sum_t \frac{P_{st}}{P_{ts}}\right)^{-1} &= \left(\sum_{t\ni E_s > E_t} \frac{1}{e^{-(E_s-E_t)/T}} + \sum_{t\ni E_s\leq E_t} \frac{e^{-(E_t-E_s)/T}}{1}\right)^{-1}\\ &= \left(\frac{\sum_t e^{-E_t/T}}{e^{-E_s/T}}\right)^{-1} = P(s)\,.\end{aligned}$$

By substituting $P(s)$ for the inverse sum in Equation 16, we get an upper bound on the perturbed probability distribution $P'(s) \;\leq\; e^{+2d\delta_m/T}P_s$. Similarly, we can show a lower bound of $e^{-2d\delta_m/T}P(s) \;\leq\; P'(s)$, yielding the final result of the theorem. □

Since SA uses the Metropolis algorithm to approximate the equilibrium distribution at all temperatures, Theorem 2 relates the behavior of SA algorithms which allow error in ΔE to that of error-free SA algorithms. This work has been presented as an analysis of error in ΔE resulting from processor interaction. It is also possible to apply the theorem to serial algorithms where the error in ΔE comes from other sources, such as an approximate calculation of ΔE, as described in [8, 10]. Preliminary versions of our results [8, 9, 10] erroneously do not include the diameter of the state space in the bounds on the probability distribution. The original work did not take into account the fact that error can accumulate as the algorithm progresses through the state space, as expressed by Inequality 13. An analysis by Greening [12] also fails to address this problem of accumulated error.

Since all macroscopic properties of a system can be derived from the distribution of states, Theorem 2 also gives bounds on average quantities such as the average energy at a given temperature:

Corollary 1 *Errors in the expectation value*

$$<\tilde{F}>=\sum_s F_s P'(s)$$

of any macroscopic property, F, of the system are bounded.

For example, the average energy in the presence of error as a function of temperature,

$$<\tilde{E}(T)>=\sum_s P'(s)\cdot E_s(T)$$

is bounded by

$$e^{-2d\delta_m/T}<E(T)>\,\leq\,<\tilde{E}(T)>\,\leq\; e^{+2d\delta_m/T}<E(T)>\;.$$

3.5 Discussion

For a given application problem and a given parallel implementation, an upper bound on the error in ΔE can be derived that depends on the number of processors and the number of moves between updates. This can be translated into bounds on $P(s)$ and $<\tilde{E}(T)>$ that also depend on architectural parameters. Given a performance model based on these same parameters, implementation choices can be linked to both the execution time and equilibrium values, providing a theoretical framework for examining the tradeoff between speed and accuracy.

This framework could profitably be extended in several ways. As with all work to date in this field, our results only apply to equilibrium conditions. They do not address the impact of error on the cooling schedule. Convergence proofs for both serial and parallel simulated annealing, of which we are aware, prove convergence to a global optimum given restrictions on the annealing schedule such that the simulated annealing algorithm will require infinitely long to complete. Yet, in practice, simulated annealing is used to obtain near-optimal solutions in finite time. Until we have a theoretical model of the finite time behavior of simulated annealing, we will not have a full understanding of its behavior in practice.

Second, an average case analysis would provide a more accurate assessment of the impact of error on simulated annealing. The bounds presented in this section show that worst-case error can be very large. One reason is that the bounds on $P'(s)$ become large as T approaches zero. This is because the worst error, δ_m, is independent of temperature. In implementations of "real world" problems, the acceptance rate decreases with T and the probability that other processors will change state during the current move also decreases. We expect that the average error $<\delta>$ will decrease with T with the result that the ratio $<\delta>/T$ will not become unbounded as the T approaches zero.

A second problem is that the bounds are not tight. First, since this is a worst case analysis, our model assumes that the worst error occurs at every move. Therefore, $e^{-\delta_m/T}$ is not a tight lower bound on perturbation in the acceptance probability, $e^{-\delta_{su}^m/T}$. Second, an error in ΔE does not necessarily lead to a wrong decision in the accept/reject decision. For example, if ΔE and $\Delta E'$ are both negative, no error will occur since the move will be accepted whether error is present or not. However, only if a wrong decision is made do errors in ΔE contribute to a perturbation in the equilibrium distribution of states. An average case analysis would address these problems also.

4 Conclusion

We have presented results concerning the efficient implementation of simulated annealing on parallel architectures. In parallel simulated annealing algorithms based on executing several moves in the state space simultaneously, a tradeoff between accuracy and efficiency occurs. The efficiency of these algorithms can be improved at the expense of accuracy by relaxing synchronization constraints. This approach has been studied experimentally, but was not well understood theoretically.

In this paper, we first introduced a formal model of parallel simulated annealing algorithms in which error occurs because some moves are scheduled simultaneously without synchronization. The error in the change in energy, ΔE, caused by these moves was expressed formally in terms of changes in state due to the actions of other processors. Second, we proved a theorem concerning the effect of errors in ΔE on systems which are brought to equilibrium using the Metropolis algorithm. These errors in the cost calculations may derive from parallelism or from other sources such as an approximate cost function. The theorem relates bounds on errors in ΔE to bounds on the resulting change in the distribution of states. Our bounds on the equilibrium state distribution can be used to compute bounds on macroscopic properties of the system, such as the average energy in the presence of error. Since simulated annealing depends on the system always being close to equilibrium, by showing how the distribution of states at equilibrium at a given temperature is affected by error in ΔE, we express the impact of error on the behavior of the simulated annealing algorithm. Finally, we discuss how our results, given a particular architecture and combinatorial optimization problem, could be used to examine how speed and accuracy are related through implementation choices. We conclude by discussing the limitations of worst-case analysis and point out how an average-case analysis would correct them.

Acknowledgements

We wish to thank Terry Boult, Scott Kirkpatrick and Greg Sorkin for carefully reading earlier versions of this paper and Ken Steiglitz and the anonymous reviewers for helpful suggestions. In particular, Greg Sorkin pointed out the importance of taking the diameter of the state space into account in Theorem 2.

References

[1] Robert Azencott. *Simulated Annealing : Parallelization Techniques.* Wiley, 1992.

[2] P. Banerjee, M. H. Jones, and J. S. Sargent. Parallel simulated annealing algorithms for cell placement on hypercube multiprocessors. *IEEE Transactions on Parallel and Distributed Systems*, 1(1):91–106, 1990.

[3] A. Casotto, F. Romeo, and A. Sangiovanni-Vincentelli. A parallel simulated annealing algorithm for the placement of macro-cells. *IEEE Transactions on Computer-Aided Design*, CAD-6(5):838–847, September 1987.

[4] A. Casotto and A. Sangiovanni-Vincentelli. Placement of standard cells using simulated annealing on the connection machine. In *International Conference on Computer-Aided Design*, pages 350–353. IEEE, November 1987.

[5] V. Czerny. Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm. *Journal of Optimization Theory and Applications*, 45:41–51, 1985.

[6] F. Darema, S. Kirkpatrick, and V. A. Norton. Parallel techniques for chip placement by simulated annealing on shared memory systems. In *International Conference on Computer Design*, pages 87–90. IEEE, 1987.

[7] M. D. Durand. Parallel simulated annealing: Accuracy vs. speed in placement. *IEEE Design and Test of Computers*, pages 8–34, June 1989.

[8] M. D. Durand. *Trading Accuracy for Speed in Parallel Simulated Annealing Algorithms.* PhD thesis, Columbia University, New York, NY, 1990.

[9] M. D. Durand and S. R. White. Permissible error in parallel simulated annealing. In *International Workshop on Placement and Routing.* Microelectronics Center of North Carolina, May 1990.

[10] M. D. Durand and Steve R. White. Permissible error in parallel simulated annealing. Technical Report RC15487, IBM, Yorktown Heights, NY, February 1990.

[11] Saul B. Gelfand and Sanjoy K. Mitter. Simulated annealing with noisy or imprecise energy measurements. *Journal of Optimization Theory and Applications*, 62(1):49–62, July 1989.

[12] D. R. Greening. Equilibrium conditions of asynchronous parallel simulated annealing. In *International Workshop on Placement and Routing.* Microelectronics Center of North Carolina, May 1990.

[13] Daniel R. Greening. Parallel simulated annealing techniques. *Physica D: Non-linear Phenomena*, 1990. (also appeared as IBM Research Report RC 15268).

[14] D.R. Greening and F. Darema. Rectangular spatial decomposition methods for parallel simulated annealing. In *Proceedings of the International Conference on Supercomputing*, Crete, Greece, June 1989.

[15] L. K. Grover. A new simulated annealing algorithm for standard cell placement. In *Proceedings of the International Conference on Computer-Aided Design*, pages 378–80, 1986.

[16] L. K. Grover. Simulated annealing using approximate calculations. In *Computer Aided VLSI Design*, volume 6. Ablex Publishing Corp, 1990. (Also appeared as ATT Bell Labs Technical Report 52231-860410-01).

[17] C.-E. Hong and B. M. McMillin. Relaxing synchronization in distributed simulated annealing. *IEEE Transactions on Parallel and Distributed Systems*, 6(2):189–195, 1995.

[18] R. Jayaraman and R. A. Rutenbar. Floorplanning by annealing on a hypercube multiprocessor. In *International Conference on Computer-Aided Design*, pages 346–349. IEEE, November 1987.

[19] Rajeev Jayaraman and Frederica Darema. Error tolerance in parallel simulated annealing techniques. In *Proceedings of the International Conference on Computer Design*, pages 545–548. IEEE Computer Society Press, 1988.

[20] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):45–54, May 1983.

[21] S. A. Kravitz and R. A. Rutenbar. Placement by simulated annealing on a multiprocessor. *IEEE Transactions on Computer-Aided Design*, CAD-6(4):534–549, July 1987.

[22] N. Metropolis, A. Rosenbluth, A. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087, 1953.

[23] F. Reif. *Fundamentals of Statistical and Thermal Physics.* McGraw-Hill, Inc., 1965.

[24] J. S. Rose, W. M. Snelgrove, and Z. G. Vranesic. Parallel standard cell placement algorithms with quality equivalent to simulated annealing. *IEEE Transactions on Computer-Aided Design*, CAD-7(3):387–396, March 1988.

[25] P.J.M. van Laarhoven and E.H.L. Aarts. *Simulated Annealing: Theory and Applications.* D. Reidel Publishing Company, Boston, 1987.

[26] C-P. Wong and R-D. Fiebrich. Simulated annealing-based circuit placement algorithm on the connection machine system. In *International Conference on Computer Design*, pages 78–82. IEEE, 1987.

This research was sponsored in part by National Science Foundation (NSF) grant no. CCR-0122581.