Revised draft - submitted for publication.

A Fast Method for Generalized Starting Temperature Determination in Monotonically Cooling Two-Stage Simulated Annealing Systems

James M. Varanelli, *Student Member, IEEE*, and James P. Cohoon, *Member, IEEE*

ABSTRACT

We propose a method for determining the starting temperature in two-stage simulated annealing systems utilizing traditional monotonically cooling temperature schedules. While most previous work in this area has focused on ad hoc experimentally-derived constant starting temperatures for the low temperature annealing phase, this paper presents a more formal method for generalized starting temperature determination for the aforementioned class of two-stage simulated annealing systems. We have tested our method on three NP-hard optimization problems using both classic and adaptive cooling schedules. The experimental results have been consistently very good—on average the running time is halved when using an adaptive cooling schedule and reduced by a third in the case of the classic schedule—with no average loss in solution quality.

I. INTRODUCTION

The *simulated annealing* (SA) algorithm [4, 20] has proven to be an effective optimization tool in the field of VLSI computer-aided design (CAD) as well as other diverse fields such as image processing and operations research [2, 4, 9, 11, 16, 22, 24, 29, 35, 39]. This stems from both its general applicability to a wide range of NP-hard combinatorial optimization problems and that it consistently produces high quality approximate solutions to these problems. SA has only one significant disadvantage—its typically very long computation times.

There has been considerable effort aimed at accelerating SA. Most of this work has concentrated on the development of faster *cooling schedules* [1, 15, 23, 24, 29], alternative move generation/acceptance strategies [8], noisy cost evaluation [9], and optimal finite-time temperature schedules [3, 12, 13, 37]. Another approach, and the one we consider here, is *two-stage simulated annealing* (TSSA) [10, 16, 33, 34]. Assuming a traditional monotonically cooling temperature schedule, a faster heuristic algorithm is used to replace the SA actions occurring at the highest temperatures in a TSSA system. The heuristic is then followed by a conventional SA approach initiated at a lower-than-normal temperature in an attempt to improve the heuristic solution. TSSA is especially beneficial for problems in which SA produces solutions of quality competitive with the most successful tailored heuristics, such as VLSI network partitioning [7, 16, 19, 22] and standard-cell placement [9, 33, 35, 39]. As will be seen in Section IV, even heuristics that produce mediocre solutions in comparison to SA can result in significant TSSA time savings over SA.

The principal consideration in the design of a traditional TSSA system is the determination of the starting temperature for the SA phase. If the chosen temperature is too low, TSSA can become prematurely trapped in some local optimum resulting in lower solution quality than standard SA. If the chosen temperature is too high, much of the structure of the first-stage heuristic solution can be wasted because of too much algorithmic hill-climbing. Early TSSA approaches [10, 16, 33] were based on finding a reasonable constant starting temperature for the SA phase. They require a significant amount of experimentation with both the chosen heuristic and the specific SA implementation being used. The primary advantage of a constant starting temperature is that once the temperature has been determined and incorporated into the TSSA system, computational overhead is very low. The obvious disadvantage is that if any of the heuristic, the SA implementation, or the problem itself changes, a new starting temperature must be found.

Rose, Klebsch, and Wolf [34] present a more gener-

Manuscript received _____; revised February 13, 1995. This work has been supported by the National Science Foundation under grants MIP-9107717 and CDA-8922545. Their support is greatly appreciated.

The authors are with the Department of Computer Science, University of Virginia, Charlottesville, VA 22903-2442.

alized method for determining the starting temperature in traditional monotonically cooling TSSA systems. Their method is based on Markov equilibrium dynamics. An approximate probability distribution for the change-in-cost function $P(\Delta C)$ is found by generating a large number of random moves from the first-stage heuristic solution. This approximation is used to locate the corresponding SA temperature in a binary-search procedure over the range $[0, t_0]$, where t_0 is the initial temperature of standard SA starting from a random solution to the given problem. At each trial temperature, the approximate $P(\Delta C)$ distribution is used to calculate the total expected cost of all positive moves and the total expected cost of all negative moves. When the magnitudes of the two values are found to be equal, the current trial temperature is returned as the starting temperature for the SA phase, since $E(\Delta C) = 0$ in SA equilibrium. Their method is shown to produce good results for the standard cell placement problem. However, there is no attempt to apply the method to other problems. Additionally, there are both problem- and formulation-dependent constraints on the choice of first-stage heuristic and a high computational cost that have discouraged its widespread adoption [34].

Analysis of these previous methods for starting temperature determination offers insight into desirable properties for a new method. The method should be generally applicable with respect to problems and traditional SA implementations; it should be relatively insensitive to the given starting solution so as to avoid constraints on the choice of the first-stage heuristic; and it should be as computationally inexpensive as possible. This paper presents a method for starting temperature determination in traditional monotonically cooling TSSA systems that meets these goals. The SA algorithm is described in Section II; Section III presents the derivation of the proposed method; Section IV presents the experimental results for three different NP-hard combinatorial optimization problems, namely the VLSI network partitioning (VLSI-NPP), rectilinear Steiner minimal tree (RSMT), and traveling salesperson (TSP) problems; and Section V discusses robustness issues.

II. SIMULATED ANNEALING

In this section we first describe the SA algorithm in general. We then present behavior characteristics of the traditional SA algorithm that will be used in the derivation of our method of starting temperature determination for traditional TSSA systems given in Section III.

2.1. The Algorithm

The SA algorithm was first introduced by Kirkpatrick, Gelatt, and Vecchi [20] and independently by Cerny [4] as a problem-independent combinatorial optimization technique. It is a generalization of the Metropolis Monte Carlo simulation [27]. It combines the advantages of iterative improvement techniques with randomizing techniques to yield a powerful optimization engine.

The traditional SA process typically starts with a random solution to the optimization problem in question. Through the use of some *generation mechanism*, a copy of the current solution is randomly perturbed to form a new solution. This new solution is subjected to the *Metropolis acceptance criterion*. The Metropolis criterion always accepts the perturbed solution as the next current solution if its cost—as defined by the given problem's *cost function*—is lower than that of the current solution (assuming minimization). It also allows for the probabilistic acceptance of higher-cost perturbed solutions as the next current solution, enabling the SA algorithm to climb out of local optima. This probabilistic acceptance is a function of the SA temperature and the difference in cost between the current and perturbed solutions.

The *initial temperature* t_0 is chosen high enough such that nearly all Metropolis trials result in acceptance. After a certain number of Metropolis trials, the temperature is lowered according to some *decrement rule*. This process continues until some *stop criterion* is met, at which point the *best-so-far* or *BSF* [3, 36] solution is returned and the algorithm is terminated. If instead the last solution visited is returned at termination, the algorithm is called *where-you-are* or *WYA*-SA. Pseudo-code for the BSF-SA algorithm using the Metropolis criterion is shown in Fig. 1.

The sequence of solutions generated at a fixed temperature can be mathematically modeled as a *homogeneous Markov chain*, due to the fact that the outcome of any given Metropolis trial depends only upon the outcome of the previous Metropolis trial [32]. Using the homogeneous Markov chain model, several sets of authors [22, 26, 29, 32] independently show that the SA algorithm as described above will converge asymptotically to a globally optimal solution given an infinite number of state transitions at each temperature and an infinite number of monotonically decreasing temperature values that in the limit approach zero. According to the theory, the temporal distribution of solutions will become *stationary* for the Markov chain executing at the current temperature, known as the *Boltzmann distribution*, after an infinite number of Metropolis trials. At this point, the Markov chain is in *equilibrium* and the temper-

```
Simulated_Annealing()
{
   initialize(i, t);
   i_BSF = i;
   do {
      do {
         j = perturb(i);
         Δc_ij = c(j) - c(i);
         if ((Δc_ij ≤ 0) || (random() < exp(-Δc_ijj/t))) {
            i = j;
            if (c(i) < c(i_BSF)) i_BSF = i;
         }
      } while (equilibrium has not been reached);
      decrement(t);
   } while (stop criterion has not been met);
   return(i_BSF);
}
```

Fig. 1: The traditional BSF-SA algorithm using the Metropolis acceptance criterion.

ature is lowered. Equilibrium must be regained at each new temperature if the optimal solutions are to be found.

Obviously, asymptotic convergence can only be approximated by any SA implementation. The majority of SA implementations proposed in the literature attempt to follow the theoretical model shown to converge asymptotically to the set of optimal solutions [1, 4, 15, 20, 23, 29, 35, 39]. The main point of commonality in these approaches is the use of monotonically decreasing temperature schedules. The differences can be traced to the way the authors choose to approximate Markov equilibrium, called *quasi-equilibrium*. The choices made in defining quasi-equilibrium for the given schedule have a direct impact on how the temperature is lowered. There are many open issues regarding the concept of quasi-equilibrium that are beyond the scope of this paper.

Recent work with optimal finite-time temperature schedules [3, 12, 13, 37] points to the possibility of non-monotone or even warming or periodic temperature schedules outperforming traditional monotonically cooling SA schedules given a finite amount of computation time. This preliminary work is encouraging, but optimal temperature scheduling lacks the wealth of theoretical and empirical results available for traditional SA. For this reason we choose not to focus on optimal finite-time temperature schedules. We instead focus on traditional SA cooling schedules that attempt to approximate the theoretical model. These types of schedules will continue to be important for many types of combinatorial applications. The automatic determination of the starting temperature in TSSA systems utilizing non-monotone optimal finite-time temperature schedules remains an open problem.

Traditional SA cooling schedules tend to fall into one of two classes depending upon the employed decrement rule—*fixed* or *adaptive*. Fixed schedules typically have decrement rules of the form $t_k = t_0 \cdot \alpha^k$, where the temperature decrement size is kept proportionately constant with $0 < \alpha < 1$. Adaptive schedules dynamically vary the size of the temperature decrements according to various aggregate statistics of each Markov chain, such as the mean and variance of the cost. The seminal papers [4, 20] both introduced fixed decrement rules, so we tend to refer to this type of schedule as the *classic schedule*. The majority of SA applications present in the literature use some variation of the classic schedule. Adaptive schedules tend to produce better quality solutions than do classic schedules at the cost of increased computation time. This can limit their practical applicability.

For the purpose of testing our proposed TSSA method, we implement one classic and one adaptive schedule for each of the problems in our test suite. The classic schedule is the one proposed by Kirkpatrick, Gelatt, and Vecchi [20]. The adaptive schedule is that of Aarts and van Laarhoven [1]. Results for the six problem/schedule combinations are presented in Section IV.

2.2. Characteristic Behavior

One way of ensuring general applicability of the proposed TSSA methodology is to base the determination of the starting temperature on some characteristic behavior of the traditional SA algorithm. Large-scale numerical studies examining solution densities at varying SA temperatures have been conducted for different pseudo-random combinatorial problems by four different sets of authors [2, 11, 28, 38]. All four independently present evidence that supports a *typical* behavior of the expected cost E_k and standard deviation σ_k with respect to SA temperature t_k given a traditional SA cooling schedule. Specifically, the investigations conclude that at all temperatures except those very close to the temperature corresponding to the optimal value of the cost function, the following behaviors can be noted:

$$E_k \approx E_\infty - \left(\sigma_\infty^2 / t_k\right); \text{ and} \tag{1}$$

$$\sigma_k \approx \sigma_\infty, \tag{2}$$

where E_∞ and σ_∞ respectively represent the expected cost and the standard deviation of the cost over the solution space. Additionally, the investigations independently show that for this same range of temperatures, the probability distribution of the cost values can be closely approximated by a normal distribution.

If we assume that the first-stage heuristic solution for a TSSA system is in equilibrium for the SA phase at some temperature, then we could use the equation shown

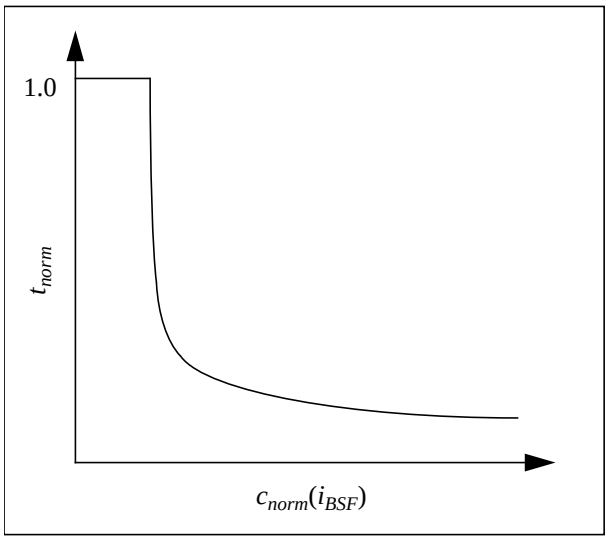

Fig. 2: Normalized BSF cost $c_{norm}(i_{BSF})$ vs. normalized SA temperature t_{norm}.

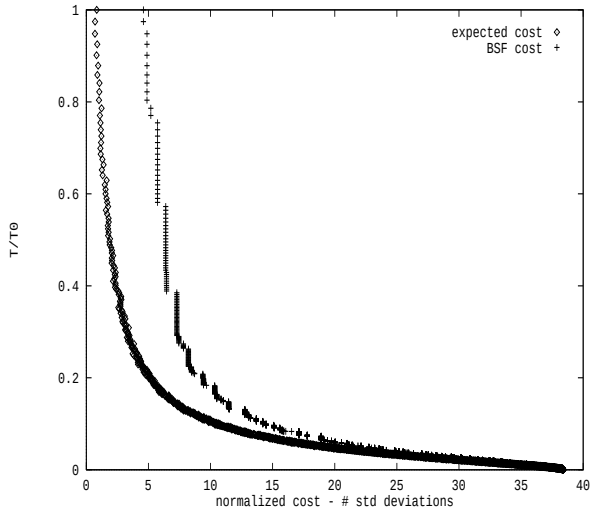

Fig. 3: Plots of normalized SA expected cost E_k and BSF cost $c_{norm}(i_{BSF})$ vs. normalized SA temperature t_{norm} produced by an actual SA run for the 318-city TSP instance.

in (1) as a reasonable temperature approximation method. However, as pointed out by Rose, Klebsch, and Wolf [34], this is almost never the case. This assumption is strictly valid only for first-stage solutions produced by the same SA algorithm while in equilibrium, a very unlikely TSSA scenario. Under this assumption, approximation accuracy would be highly problem-dependent, and could result in a degradation of solution quality for TSSA as compared to standard SA. For this reason, we assume that the first-stage heuristic solution is the BSF solution at some SA temperature, rather than the equilibrium solution.

The characteristic behavior of the BSF cost over the course of the traditional SA algorithm has not been well documented in the literature. Sibani, Pedersen, Hoffman and Salamon [36] present a scaling method to estimate the global minima of NP-hard combinatorial problems during a SA run based on the E_{BSF} distribution, but make no attempt to describe the evolutionary behavior of the BSF cost with respect to decreasing temperature. Fig. 2 illustrates the characteristic SA BSF curve that results from plotting the evolution of the normalized BSF cost $c_{norm}(i_{BSF})$ against normalized temperature t_{norm}, using the normalizations $c_{norm}(i) = (E_\infty - c(i))/\sigma_\infty$ and $t_{norm} = t_k/t_0$. Similar plots from actual SA runs can be seen in Figs. 3, 5, and 6.

Fig. 3 shows a comparison of BSF cost evolution against expected cost evolution from an actual SA run. As can be seen in the figure, the BSF curve is shifted to the right of the expected cost curve, indicating that the expected cost E_k at temperature t_k is some number of standard deviation units σ_k greater than the BSF cost. In the next section, we present a method to probabilistically calculate this difference in expected and BSF cost, to be used in our method of associating a SA temperature with a given solution assumed to be the BSF solution.

III. DERIVATIONS

Given the behavioral information presented in the previous section, we now present the following two propositions that describe the derivation of our TSSA starting temperature determination methodology.

Proposition 1: Given a solution i for some combinatorial optimization problem, the following function can be used to closely approximate the SA temperature $t_k(i)$ at which i would be found as the BSF solution:

$$t_k(i) \approx \frac{\sigma_\infty^2}{E_\infty - c(i) - \gamma_\infty \sigma_\infty}. \quad (3)$$

Proof: We will use the behavior of the expected cost E_k represented by (1) as a starting point. Solving (1) with respect to absolute temperature t_k gives:

$$t_k \approx \frac{\sigma_\infty^2}{E_\infty - E_k}. \quad (4)$$

Since we are assuming that the given solution i corresponds to the SA BSF solution as described in the previous section, we cannot reliably use (4) as a temperature approximation. A function relating E_k and $c(i_{BSF})$ is necessary to complete the derivation. If we assume i_{kmin} is the minimum-cost solution seen during the k^{th} Markov chain executing at temperature t_k, then we know that the

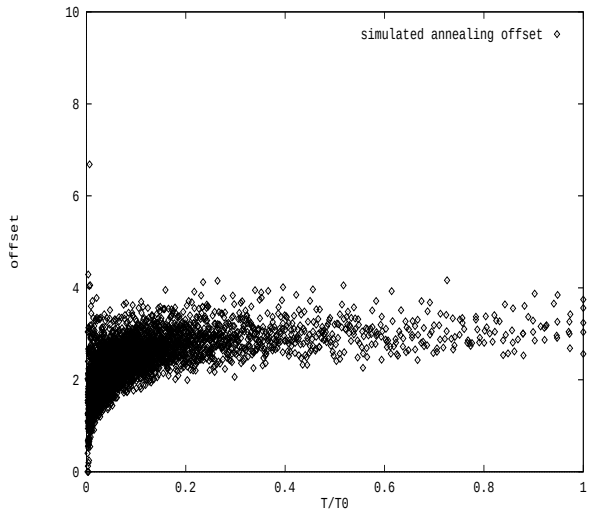

Fig. 4: Evolution of the offset γ_k for the 100-city TSP instance.

following relation holds:

$$E_k \geq c(i_{kmin}) \ . \tag{5}$$

This implies that E_k is some number of standard deviation units σ_k greater than $c(i_{kmin})$ at temperature t_k. We call this number the *offset* and denote it by the symbol γ_k. In this context, (5) can be expressed as:

$$E_k = c(i_{kmin}) + \gamma_k \sigma_k . \tag{6}$$

We can replace the σ_k term in (6) with σ_∞ as described by (2). We can also approximate i_{kmin} with i_{BSF} to give:

$$E_k \approx c(i_{BSF}) + \gamma_k \sigma_\infty \ . \tag{7}$$

Using the behavior of the standard deviation σ_k described by (2), combined with the assumption that the solution density is nearly normally distributed, we expect the offset γ_k to remain approximately constant at the higher temperatures while converging quickly towards zero close to the temperature corresponding to the optimal value of the cost function. Our experimental evidence indeed supports this behavior. This can be seen graphically in Fig. 4, showing the evolution of the offset over the course of the SA algorithm for a number of runs. This behavior allows us to closely approximate the γ_k term in (7) with γ_∞. Replacing the E_k term in (4) by the updated version of (7) gives us the following:

$$t_k(i_{BSF}) \approx \frac{\sigma_\infty^2}{E_\infty - c(i_{BSF}) - \gamma_\infty \sigma_\infty} \ . \tag{8}$$

Finally, instantiating (8) with the first-stage heuristic solution i for i_{BSF} gives (3).

■

Equation (3) serves as the basis for our proposed method of starting temperature determination in traditional TSSA systems. However, the offset γ_∞ is still an unknown. It is important to note that without the offset term, (3) generates temperature approximations that could possibly be too low depending upon the problem being solved, resulting in TSSA solution quality less than that of standard SA. This is discussed in more detail in Section 5.2. The following proposition describes the calculation of γ_∞.

Proposition 2: Given a SA formulation for some combinatorial optimization problem with Markov chain length L_M, the offset γ_∞ can be calculated probabilistically with the equation:

$$P[E_\infty - \gamma_\infty \sigma_\infty < X < E_\infty + \gamma_\infty \sigma_\infty] \approx 1 - |L_M|^{-1} \tag{9}$$

Proof: As discussed in the previous section, we make use of the observation that we can use a normal distribution to closely approximate the probability distribution of the cost values over each Markov chain. Using this assumption, the cost values seen over the course of a Markov chain can be represented by a normally distributed random variable X with probability distribution function:

$$f(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\left[\frac{x-\mu}{\sigma}\right]^2/2} . \tag{10}$$

A simple change in variable leads to the following cumulative distribution function (CDF):

$$\Phi(z) = \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt \, , \tag{11}$$

where $t = (x - \mu)/\sigma$ and $dx = \sigma\, dt$. This CDF can then be used in such a way as to determine the probability of generating a solution with a given cost. Explicitly, this can be written as

$$P[X \leq x] = \Phi\left(\frac{x-\mu}{\sigma}\right). \tag{12}$$

If we let $z = (x - \mu)/\sigma$, the term $z_{\rho/2}$ determines the percentile ρ of the normal distribution, $\Phi(z_{\rho/2}) = \rho$, to which $X = x$ is likely to belong. For our purposes, it is useful to consider these normal probabilities in terms of standard deviation units away from the mean. In this context, $z_{\rho/2}$ can be used in the following manner:

$$P[\mu - z_{\rho/2}\sigma < X < \mu + z_{\rho/2}\sigma] = 1 - \rho \, . \tag{13}$$

It should be clear that the $z_{\rho/2}$ term in the above equation corresponds to the offset γ_k used in our starting temperature determination method. The problem remains to calculate appropriate ρ values and their corresponding $z_{\rho/2}$ values to be used in determining the expected value for γ_∞. Using the behavior characteristics described earlier in this section, we can use the offset value γ_1 for the first Markov chain to obtain a reasonable approximation for γ_∞. We need now only determine the expected offset for the first Markov chain.

Since virtually all L_M transitions will be accepted

during the first Markov chain of the SA cooling schedule, the stationary distribution of this chain is uniform over the state space [2, 11, 28, 38]. This implies that the mean μ_1 and standard deviation σ_1 of the cost values seen over the first Markov chain will approach E_∞ and σ_∞ respectively. Using this fact, we can assume that each generated solution has a probability very close to $|L_M|^{-1}$ of having the minimum cost value seen over the course of the first Markov chain. This will serve as the value for ρ. Conversely, each generated solution will have a probability very close to 1 - ρ of having a cost greater than that of the minimum-cost value seen over the course of the first Markov chain. Using these values, we can now determine $z_{\rho/2}$, and hence our offset γ_∞. Since we are considering the first Markov chain, we can substitute E_∞ for μ and σ_∞ for σ in (13), giving us (9).

■

Given the appropriate value for ρ, the corresponding $z_{\rho/2}$ value can be calculated via numerical methods or by a table lookup. Tables I-III show results for approximating the offset of each problem instance in our test suite using table values. Results for each instance are averaged over 20 runs. As can be seen in the tables, computed $z_{\rho/2}$ values match very closely with observed average offsets $\bar{\gamma}_\infty$.The variance seen in the offset values generated for any particular problem instance is generally quite high. However, in practice, this did not affect the ability of the TSSA systems to converge to the same average quality solutions as the corresponding standard SA algorithms for the same instances. This can be seen in the TSSA results presented in Section IV.

In addition to being required for our starting temperature determination method, the offset γ_∞ can be used as a gauge in the choice of first-stage heuristic. Intuitively we know that if the normalized cost of the first-stage heuristic solution $c_{norm}(i_{heur})$ is less than γ_∞ then TSSA will most likely not be beneficial, since this is the amount of optimization expected to occur during the Markov chain executing at the first SA temperature. This rough guideline can be further refined if we keep in mind that TSSA can only be beneficial if the cost of the first-stage solution is such that

$$t_k(i_{heur}) \approx \frac{\sigma_\infty^2}{E_\infty - c(i_{heur}) - \gamma_\infty \sigma_\infty} < t_0 . \qquad (14)$$

This implies that for TSSA to be beneficial, a first-stage heuristic should return solutions of absolute cost

$$c(i_{heur}) < E_\infty - \left(\frac{\sigma_\infty}{t_0} + \gamma_\infty \right) \sigma_\infty . \qquad (15)$$

Based on the above discussion, our method can be summarized in the following steps, assuming TSSA has been judged beneficial by (15):

- Execute the heuristic to obtain $c(i_{BSF})$.
- Obtain values for E_∞, σ_∞, and γ_∞.
- Use $c(i_{heur})$, E_∞, σ_∞, and γ_∞ in (5) to obtain the starting temperature approximation t_{app}.
- Set $t = t_{app}$ and begin the SA phase.

As can be seen in Figs. 5 and 6, our method produces approximations that are quite close to actual SA temperatures associated with the BSF solution for different

cells	L_M	1 - $\|L_M\|^{-1}$	$z_{\rho/2}$ (computed)	$\bar{\gamma}_\infty$ (observed)
100	100	0.9900	2.58	2.76
500	500	0.9980	3.09	3.02
833	833	0.9988	3.24	3.13
1500	1500	0.9993	3.41	3.28
3014	3014	0.9995	3.59	3.64
10000	10000	0.9998	3.87	3.88

Table I: Experimental results for approximating the offset γ_∞ for several VLSI-NPP instances.

terminals	L_M	1 - $\|L_M\|^{-1}$	$z_{\rho/2}$ (computed)	$\bar{\gamma}_\infty$ (observed)
13	47	0.9787	2.30	2.18
16	83	0.9880	2.51	2.87
20	380	0.9974	3.01	2.90
23	313	0.9968	2.95	3.01
30	695	0.9986	3.19	3.08
37	438	0.9977	3.05	3.09

Table II: Experimental results for approximating the offset γ_∞ for several RSMT instances.

cities	L_M	1 - $\|L_M\|^{-1}$	$z_{\rho/2}$ (computed)	$\bar{\gamma}_\infty$ (observed)
42	861	0.9988	3.25	3.33
50	1225	0.9992	3.35	3.31
57	1596	0.9994	3.42	3.62
100	4950	0.9998	3.72	3.55
318	50403	1.0000	4.27	4.38
532	141246	1.0000	4.45	4.56

Table III: Experimental results for approximating the offset γ_∞ for several TSP instances.

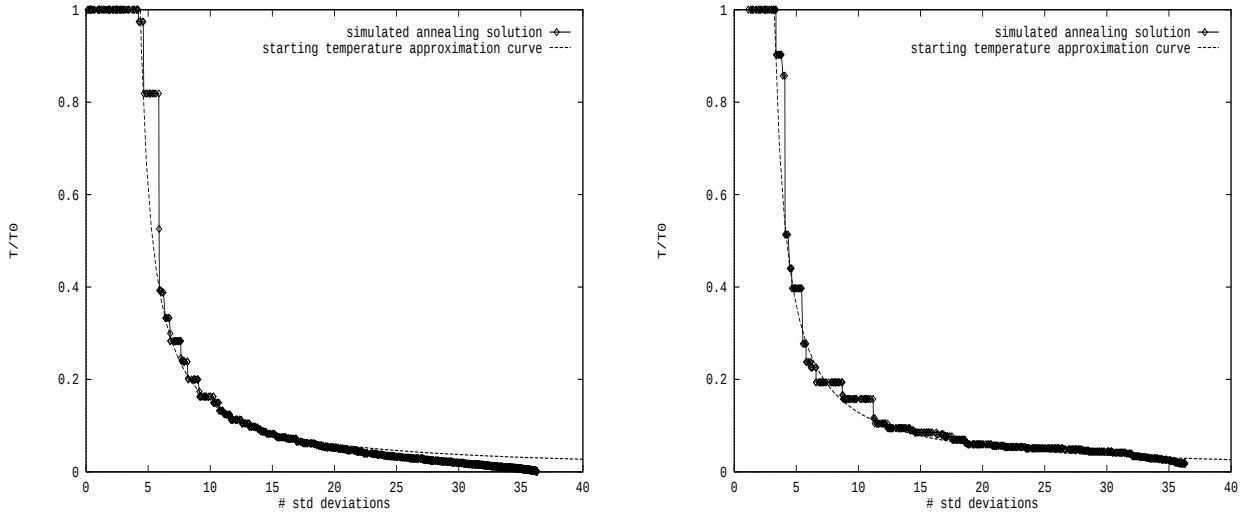

Fig. 5: Starting temperature approximation curve vs. SA BSF curves for PrimarySC1 VLSI-NPP instance concerning (a) the adaptive schedule; and (b) the classic schedule.

problem/schedule combinations. Fig. 5 shows our approximation curve plotted against actual SA BSF curves for the PrimarySC1 VLSI-NPP instance. Fig. 6 shows our approximation curve plotted against actual SA BSF curves for the 318-city TSP instance. For both figures, plots (a) and (b) concern respectively the adaptive and classic schedules. The experimental results presented in the next section indicate that in practice there is a significant time reduction seen for TSSA systems incorporating the above methodology over standard SA with no loss in solution quality.

IV. TSSA EXPERIMENTAL RESULTS

We now present results for our TSSA method applied to the VLSI-NPP, RSMT, and TSP. A short description of each problem is given in the corresponding subsection. All TSSA systems discussed here are implemented in the C/C++ programming language using the Gnu g++ compiler. All test runs were executed on a Sun SparcServer 10/51 operating under SunOS 4.1.3 (UNIX) with 128 MB of RAM. All results are averaged over 20 runs. Further discussion of these results can be found in Section V.

As mentioned in Section I, each problem is solved with two different cooling schedules—a classic schedule [20] and an adaptive schedule [1]. There are a number parameters particular to each schedule implementation, as well as some that are schedule-independent. The schedule-dependent parameters were set to their recommended values as discussed in each of the original papers. Initial temperature for both schedules is set equal to σ_∞ as recommended by Otten and van Ginneken [29] and White [38]; and the Markov chain length for both schedules is set equal to the size of the SA neighborhoods as recommended by Aarts and van Laarhoven [1].

4.1. VLSI Network Partitioning Problem

The VLSI-NPP is a generalization of the graph partitioning problem (GPP) referred to as *hypergraph partitioning* [7, 16, 19]. The input to the VLSI-NPP consists of a set of VLSI circuit elements, or *cells*, connected by a set of *nets*. Each cell has an associated positive-valued area. Each net is a *hyperedge*, and represents at least two cells that are electrically interconnected. The goal of the VLSI-NPP is to divide the cells into two blocks so as to minimize the number of nets that have cells in both blocks under the constraint that the sums of the areas of the cells in each block are approximately equal. The specified balance tolerance for each block is the size of the largest cell in the circuit *SMAX* [7]. The difference in area between the two blocks is used as a penalty term. Hence, the objective function to be minimized is

$$c(i) = |E_{cut}| + \lambda \cdot \left(\sum_{a_j \in A} area(a_j) - \sum_{b_j \in B} area(b_j) \right)^2, \quad (16)$$

where E_{cut} is the number of nets with cells in both blocks and λ is a weighting constant. For our VLSI-NPP implementation, $\lambda = 1/(4 \cdot SMAX^2)$.

The TSSA VLSI-NPP system incorporates the Fiduccia and Mattheyses (F-M) heuristic [7]. The F-M

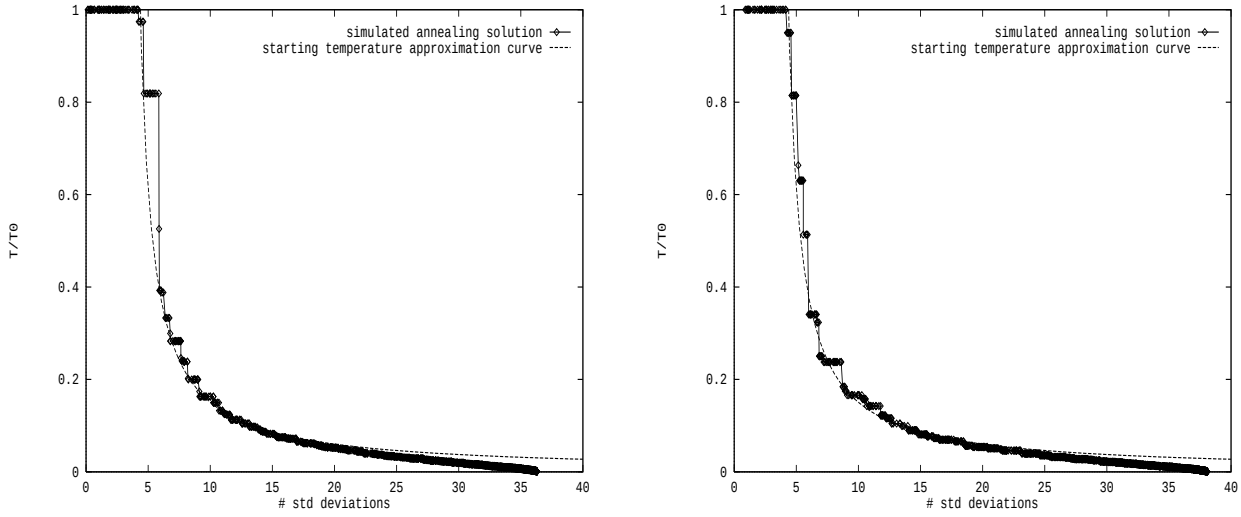

Fig. 6: Starting temperature approximation curve vs. SA BSF curves for 318-city TSP instance concerning (a) the adaptive schedule; and (b) the classic schedule.

cells	$\overline{CPU}_{SA}$ (sec)	$\overline{C}_{SA}$	$\overline{CPU}_{TSSA}$ (sec)	$\overline{C}_{TSSA}$	$\overline{\Delta}_{CPU}$%
100	0.20	16.0	.11	15.8	45.0
500	2.90	101.8	1.01	102.6	65.2
833	6.56	72.5	2.26	76.2	65.5
1500	18.72	289.9	5.95	287.3	68.2
3014	47.38	226.0	16.30	245.2	65.6
10000	324.51	2181.6	141.33	2127.8	56.4

Table IV: Results for the TSSA VLSI-NPP system using an adaptive schedule.

cells	$\overline{CPU}_{SA}$ (sec)	$\overline{C}_{SA}$	$\overline{CPU}_{TSSA}$ (sec)	$\overline{C}_{TSSA}$	$\overline{\Delta}_{CPU}$%
100	0.09	16.8	0.07	15.9	22.2
500	0.99	109.5	0.65	103.3	34.3
833	2.09	99.8	1.33	129.3	36.4
1500	4.82	320.4	2.84	318.5	41.1
3014	12.15	396.0	5.38	414.2	55.7
10000	61.45	2121.0	35.70	2108.3	41.9

Table V: Results for a TSSA VLSI-NPP system using a classic schedule.

heuristic is a generalization of the Kernighan-Lin graph bipartitioning heuristic [19]. The heuristic was selected due to its fast running times and quality of solution. The complexity of the algorithm is linear in the total number of pins, where a *pin* is an interconnection point on a cell for a particular net. We chose to only use one pass of the algorithm, since the majority of improvement takes place during the first pass. Experimental results show that SA improves solutions generated by one pass of F-M by an average of 15% relative to E_∞ in terms of standard deviation units σ_∞.

Experimental data used for evaluating the TSSA VLSI-NPP system are the SIGDA standard cell benchmark circuits PrimarySC1 (833 cells, 904 nets) and PrimarySC2 (3014 cells, 3029 nets) [31] as well as four randomly generated networks with average edge degrees and net-size distributions similar to the benchmark circuits. The randomly generated networks range in size from 100 nets with 100 cells to 10000 nets with 10000 cells. The results are given in Tables IV and V respectively for the adaptive and classic schedules. The following conventions are used for the tables: $\overline{CPU}_{SA}$ and $\overline{CPU}_{TSSA}$ represent the average CPU times respectively of SA and TSSA; $\overline{C}_{SA}$ and $\overline{C}_{TSSA}$ represent the average final BSF cost respectively for SA and TSSA; and $\overline{\Delta}_{CPU}$% represents the average percentage difference in CPU time between SA and TSSA for the given instance. These results are for flat, non-clustered partitioning. Many authors have demonstrated the fact that a clustered partitioning phase before a flat partitioning can significantly improve solution quality. This was not examined

here, but we expect the proposed TSSA methodology to behave similarly in this case. As can be seen from the tables, significant speedup is observed in the TSSA systems incorporating our method of starting temperature determination over standard SA with no average loss in solution quality. The average speed-up was approximately 61% and 39% respectively for the adaptive and classic schedules.

4.2. Rectilinear Steiner Minimal Tree Problem

The input to the RSMT problem consists of a set of n points in a plane, called *terminals*. The goal of the RSMT problem is to connect the terminals with horizontal and vertical line segments such that the sum of the lengths of the segments is minimized. The connected terminals should form an acyclic tree such that all of the terminals serve as endpoints to various segments. Additional points, called *Steiner points*, can also be used to connect the terminals. Using a result of Hanan [12], we can restrict a search for an optimal solution to Steiner point locations that lie on a grid imposed by the terminals. This grid defines at most $O(n^2)$ possible Steiner locations. Hanan's result also allows us to limit the actual number of Steiner points to at most n - 1. The objective function is simply

$$c(i) = \sum_{e_j \in E} length(e_j), \tag{17}$$

where E is the set of all edges in the tree.

The heuristic chosen for the first phase of the TSSA RSMT system is based on Kruskal's minimum-spanning tree algorithm [21]. Kruskal's algorithm is first run to obtain the minimum-spanning tree for the n terminals using no Steiner points. Then up to n - 1 Steiner points are added in a greedy fashion from the set of possible Steiner locations to form the initial solution for the SA phase. SA improves solutions produced by our variation of Kruskal's algorithm by an average of 10% relative to E_∞ in terms of standard deviation units σ_∞.

Experimental data used for evaluating the TSSA RSMT system consists of four of the larger nets from the SIGDA benchmark circuit PrimarySC2 as well as randomly generated 20 and 30 terminal networks. The nets taken from PrimarySC2 range in size from 13 to 37 terminals. The placements of the nets from PrimarySC2 are intermediate solutions generated by a local placement and routing package. The results are shown in Tables VI and VII. Again, significant speedup is noted for the TSSA RSMT system over standard SA with no loss in solution quality. The average speedup was approximately 41% and 30% respectively for the adaptive and classic schedules.

terminals	$\overline{CPU}_{SA}$ (sec)	$\overline{C}_{SA}$	$\overline{CPU}_{TSSA}$ (sec)	$\overline{C}_{TSSA}$	$\overline{\Delta}_{CPU}$%
13	11.96	1964.3	5.95	1958.4	50.3
16	54.10	3022.8	23.45	3004.2	56.7
20	921.58	304.5	669.55	304.4	27.3
23	1144.48	7550.0	711.76	7553.4	37.8
30	6863.57	359.3	4191.14	359.3	38.9
37	5720.76	6490.0	3648.04	6491.0	36.2

Table VI: Results for a TSSA RSMT system using an adaptive schedule.

terminals	$\overline{CPU}_{SA}$ (sec)	$\overline{C}_{SA}$	$\overline{CPU}_{TSSA}$ (sec)	$\overline{C}_{TSSA}$	$\overline{\Delta}_{CPU}$%
13	8.67	1949.5	6.05	1953.0	30.2
16	32.59	3005.6	22.23	2998.8	31.8
20	274.38	305.4	215.50	305.0	21.5
23	373.03	7556.4	264.66	7553.2	29.1
30	1512.38	360.5	1169.05	361.5	22.7
37	1628.60	6490.0	1109.23	6494.8	31.9

Table VII: Results for a TSSA RSMT system using a classic schedule.

4.3. Traveling Salesperson Problem

The input to the TSP consists of a symmetric $n \times n$ distance matrix d, representing distances between n cities. The goal is to find a minimum-length tour that visits each city exactly once while terminating at the city of origin. The objective function to be minimized is

$$c(i) = \sum_{j}^{n-1} d_{j,j+1} + d_{n,1} \ . \tag{18}$$

For the first stage of the TSSA TSP system, a variation of the Croes heuristic [5] is used. SA improves solutions produced by our variation of the Croes algorithm by an average of 10% relative to E_∞ in terms of standard deviation units σ_∞.

Experimental data used for evaluating the TSSA TSP system consists of the following instances: the 42 city problem of Dantzig, Fulkerson, and Johnson [6]; a randomly generated 50 city problem; the 57 city problem of Karg and Thompson [18]; a randomly generated 100 city problem; the 318 city problem of Lin and Kernighan [25]; and the 532 city problem of Padberg and Rinaldi

[30]. The results are given in Tables VIII and IX. As with the previous two problems, significant speedup is noted for the TSSA TSP system over standard SA with no loss in solution quality. The average speedup was approximately 48% and 35% respectively for the adaptive and classic schedules.

In summary, the average running time speedup for TSSA over SA for all problem instances in the test suite is approximately 50% when using an adaptive cooling schedule and approximately 35% when using a classic cooling schedule. Both figures are clearly significant. Equally important is the fact that the TSSA systems are on average able to produce the same quality solutions as the corresponding single-stage SA approaches.

cities	$\overline{CPU}_{SA}$ (sec)	$\overline{C}_{SA}$	$\overline{CPU}_{TSSA}$ (sec)	$\overline{C}_{TSSA}$	$\overline{\Delta}_{CPU}\%$
42	4.46	704.4	3.14	703.4	29.6
50	7.02	242.9	2.98	236.4	51.5
57	11.40	13086.4	4.73	13133.0	53.7
100	51.23	313.2	21.53	307.3	54.9
318	1722.98	42869.6	703.22	42835.0	59.0
532	10104.39	88105.5	6044.17	88162.1	40.2

Table VIII: Results for the TSSA TSP system using an adaptive schedule.

cities	$\overline{CPU}_{SA}$ (sec)	$\overline{C}_{SA}$	$\overline{CPU}_{TSSA}$ (sec)	$\overline{C}_{TSSA}$	$\overline{\Delta}_{CPU}\%$
42	0.90	705.2	0.63	704.6	30.0
50	1.23	245.2	0.85	243.4	30.9
57	1.92	13140.7	1.29	13106.5	32.8
100	6.71	324.9	4.23	322.3	37.0
318	125.98	43295.2	87.63	43375.1	30.4
532	651.70	89555.9	345.49	89648.6	47.0

Table IX: Results for a TSSA TSP system using a classic schedule.

V. ROBUSTNESS OF METHODOLOGY

An important aspect of the methodology that has not been addressed is that of robustness. Specifically, are there any constraints on the choice of the first-stage heuristic? Also, what happens if we assume that the first-stage heuristic solution corresponds to the equilibrium solution at some temperature t_k as opposed to the BSF solution? Additionally, what happens in the case of schedule abbreviation, where the highest temperatures are found to be superfluous and a lower SA initial temperature is used? These issues are examined in the following three subsections.

5.1. First-stage Heuristics with Good Solution Quality

The experiments of Section IV focus on the use of first-stage heuristics that have low computational cost and mediocre solution quality as compared to SA. This was done to illustrate the fact that even these types of simple heuristics can produce a significant time savings for TSSA over standard SA while maintaining equivalent solution quality. The quality of these first-stage solutions corresponds to the middle section of the characteristic BSF curve shown in Figs. 5 and 6. However, what happens in the case of a first-stage heuristic that produces solutions near the tail of the BSF curve, in the regime of good solutions?

In an attempt to address this question, we examined one instance of each problem from our test suite with a TSSA system incorporating a first-stage heuristic that produces high-quality solutions in relation to SA. For the TSSA VLSI-NPP, we use a full-pass F-M implementation instead of a single pass as the first stage followed by

instance	C_{opt}	C_{heur}	$\overline{\Delta}_{opt}\%$	$\overline{CPU}_{heur}$ (sec)	$\overline{C}_{TSSA}$	$\overline{\Delta}_{opt}\%$	$\overline{CPU}_{TSSA}$ (sec)	$\overline{\Delta}_{CPU}\%$
3014-cell VLSI-NPP	160	473.2	195.8	13.97	247.4	40.9	19.48	58.9
30-terminal RSMT	359	366	1.9	52.60	361.2	0.6	452.91	70.1
318-city TSP	41345	43389.8	4.9	123.33	42842.4	3.7	933.11	45.8

Table X: TSSA results with good first-stage heuristic solution quality.

adaptive SA. For the TSSA RSMT, we use the iterative 1-Steiner heuristic of Kahng and Robins [17] as the first stage followed by classic SA. Finally for the TSP, we use classic SA as the first stage followed by adaptive SA. Each of these TSSA systems was executed 20 times for the given instance. These results are presented in Table X. The optimal cost for each instance is used as a control[1]. As can be seen in the table, the proposed methodology performs just as well with heuristics that produce high-quality solutions. However, these results are not surprising in light of the fact that the approximation curve produced by our methodology overestimates temperature in the regime of good solutions, as can be seen in Figs. 5 and 6.

5.2. Expected Cost vs. BSF Cost

As was mentioned in Section 2.2, we assume that the solution returned by the first-stage heuristic corresponds to the BSF solution at some SA temperature t_k. What happens if we change this assumption such that the first-stage heuristic solution instead corresponds to the expected cost at some SA temperature t_k? A slight decrease in computation time can be expected due to the lower temperature approximations generated by the expected cost model. However, is there any difference in TSSA solution quality between the two models?

As was pointed out in Section 2.2, using the expected cost assumption could result in varying TSSA model approximation accuracy under different problems. We did not attempt to answer the question in the general case, but instead focused on the PrimarySC1 and PrimarySC2 instances of the VLSI-NPP using the two different TSSA models to examine the effect of changing the cost assumption. Using the classic SA schedule, we ran 1000 trials each of full-regime SA, TSSA with our proposed methodology under the BSF cost model, and TSSA under the expected cost model for the two instances. These results are summarized in Table XI. As can be seen in the table, TSSA under the BSF model produces solutions almost indistinguishable in cost from standard SA solutions. However, TSSA under the expected cost model generates solutions slightly greater than both standard SA and TSSA under the BSF model. Although as expected there is a slight decrease in computation time over TSSA under the BSF model, for this problem it appears that TSSA under the expected cost model results in premature freezing.

[1]The optimal cut value for the PrimarySC2 VLSI-NPP instance does not appear in the literature and is most likely not known. The given value is the best result we've obtained over thousands of runs with the adaptive SA schedule. This is for flat, non-clustered partitioning with a balance tolerance of the size of the largest cell in the circuit, or +/- 0.34% in the case of PrimarySC2. The figure is consistent with similar partitioning results presented in the literature.

5.3. Lowering the Initial SA Temperature

As was demonstrated by Johnson, Aragon, McGeoch and Schevon [16], traditional SA cooling schedules could be wasting time at the highest temperatures. Their experiments concluded that for the GPP, an abbreviated version of the classic SA schedule with one-half the usual number of temperature steps could produce solutions of equal quality to those produced by a full SA temperature regime in roughly half the computation time. How does this type of situation impact the performance of the proposed TSSA model?

We examined this question with the PrimarySC1 and PrimarySC2 instances of the VLSI-NPP. Our experiments show that we can start abbreviated SA at approximately one-third the original initial temperature and still achieve solution quality similar to full-regime SA. We ran 1000 trials of abbreviated classic SA for PrimarySC1 and PrimarySC2 with average final costs of 99.0 and 396.4 respectively, as compared to 99.8 and 396.0 for full-regime SA. The BSF curve for the abbreviated SA quickly regains the expected behavior pattern exhibited for full-regime SA as described in Section 2.2. This can be seen graphically in Fig. 7.

This would indicate that the proposed TSSA methodology will be effective with abbreviated SA if the first-stage heuristic is able to generate solutions that correspond to the remaining part of the abbreviated BSF curve. Since the chosen first-stage TSSA VLSI-NPP

instance	*Standard SA*			*TSSA w/ BSF model*			*TSSA w/ expected cost model*		
	$\overline{C}$	C_{best}	$\overline{CPU}$	$\overline{C}$	C_{best}	*CPU*	$\overline{C}$	C_{best}	$\overline{CPU}$
PrSC1	99.8	72	2.13	100.6	73	1.30	104.1	75	1.27
PsSC2	395.4	344	12.09	410.9	344	5.36	422.2	348	5.02

Table XI: Comparison of two TSSA models on PrimarySC1 VLSI-NPP instance.

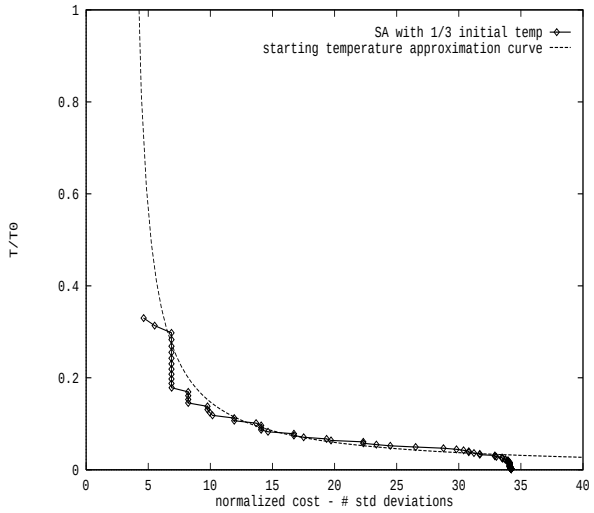

Fig. 7: Starting temperature approximation curve vs. abbreviated SA BSF curve for PrimarySC1.

heuristic generates solutions well past this point on the BSF curve, our experiments show that the proposed TSSA methodology performance for abbreviated SA is consistent with that for full-regime SA, although the percentage decrease in CPU times is lower in the abbreviated case. As is the case with full-regime TSSA, a candidate first-stage heuristic for abbreviated TSSA can be evaluated using (15) by simply replacing the t_0 term with the starting temperature for the abbreviated schedule. For this VLSI-NPP example, we replace $t_0 = \sigma_\infty$—the initial starting temperature of SA—with $t_0 = 0.33 \cdot \sigma_\infty$ to get the first-stage heuristic criterion $c(i_{heur}) < E_\infty - (3 + \gamma_\infty)\,\sigma_\infty$ for TSSA to be beneficial.

VI. CONCLUSIONS

We propose a TSSA method with a more formal basis for determining the temperature at which to begin the low temperature SA phase. The method is compatible with TSSA systems utilizing a traditional monotonically cooling temperature schedule. We have tested our method on three important optimization problems using both classic and adaptive schedules. The results have been consistently very good. On average for a SA algorithm using an adaptive cooling schedule the running time is cut in half; while for a SA algorithm using a classic schedule, the running time is reduced by one-third. Equally important is that there is on average no loss in solution quality. The TSSA methodology is also shown to be robust with respect to different first-stage heuristics and traditional SA formulations.

VII. ACKNOWLEDGEMENTS

The authors wish to thank Joe Ganley for the use of his iterative 1-Steiner RSMT implementation, as well as his insights regarding the RSMT problem in general. The authors also wish to acknowledge the comments of the expert reviewers. Their input is greatly appreciated.

VIII. REFERENCES

[1] E.H.L. Aarts and P.J.M. van Laarhoven, "A New Polynomial-Time Cooling Schedule," *Proc. IEEE ICCAD-85*, Santa Clara, CA, 206-208, 1985.

[2] E.H.L. Aarts, J.H.M. Korst, and P.J.M. van Laarhoven, "Solving Traveling Salesman Problems by Simulated Annealing," *J. Stat. Phys.*, vol. 50, 187-206, 1988.

[3] K.D. Boese and A.B. Kahng, "Best-So-far vs. Where-You-Are: Implications for Optimal Finite-Time Annealing," *Sys. and Control Letters*, vol. 22, 71-78, 1994.

[4] V. Cerny, "Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm," *J. Opt. Thry. and Appl.*, vol. 45, 41-51, 1985.

[5] G.A. Croes, "A Method for Solving Traveling-Salesman Problems," *Operations Research*, vol. 5, 791-812, 1958.

[6] G.B. Dantzig, D.R. Fulkerson, and S.M. Johnson, "Solution of a Large Scale Traveling-Salesman Problem," *Operations Research*, vol. 2, 393-410, 1954.

[7] C.M. Fiduccia and R.M. Mattheyses, "A Linear-Time Heuristic for Improving Network Partitions," *Proc. 19th ACM/IEEE DAC*, Las Vegas, NV, 241-247, 1982.

[8] J.W. Greene and K.J. Supowit, "Simulated Annealing Without Rejected Moves," *IEEE Trans. CADICS*, vol. 5, 221-228, 1986.

[9] L.K. Grover, "A New Simulated Annealing Algorithm for Standard Cell Placement," *Proc. IEEE ICCAD-86*, Santa Clara, CA, 378-380, 1986.

[10] L.K. Grover, "Standard Cell Placement Using Simulated Sintering," *Proc. 24th ACM/IEEE DAC*, Miami Beach, FL, 56-59, 1987.

[11] B. Hajek, "A Tutorial Survey of Theory and Applications of Simulated Annealing," *Proc. 24th IEEE Conf. Decision and Control*, Ft. Lauderdale, FL, 755-760, 1985.

[12] B Hajek, "Cooling Schedules for Optimal Annealing," *Math. Op. Res.*, vol. 13, 311-329, 1988.

[13] B. Hajek and G. Sasaki, "Simulated Annealing - To Cool or Not," *Sys. and Control Letters*, vol. 12, 443-447, 1989.

[14] M. Hanan, "On Steiner's Problem With Rectilinear Distance," *SIAM J. Appl. Math.*, vol. 14, 255-265, 1966.

[15] M.D. Huang, F. Romeo, and A. Sangiovanni-Vincentelli, "An Efficient General Cooling Schedule for Simulated Annealing," *Proc. IEEE ICCAD-86*, Santa Clara, CA, 381-384, 1986.

[16] D.S. Johnson, C.R. Aragon, L.A. McGeoch, and C. Schevon, "Optimization by Simulated Annealing: An Experimental Evaluation; Part 1, Graph Partitioning," *Operations Research*, vol. 37, 865-892, 1989.

[17] A.B. Kahng and G. Robins, "A New Class of Iterative Steiner Tree Heuristics with Good Performance," *IEEE Trans. CADICS*, vol. 11, 893-902, 1992.

[18] R.L. Karg and G.L. Thompson, "A Heuristic Approach to Solving Traveling-Salesman Problems," *Management Science*, vol. 10, 225-247, 1964.

[19] B.W. Kernighan and S. Lin, "An Efficient Heuristic Procedure

for Partitioning Graphs," *Bell Sys. Tech. J.*, vol. 49, 291-307, 1970.

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

[21] J.B. Kruskal, "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem," *Proc. American Math. Soc.*, vol. 7, 48-50, 1956.

[22] P.J.M. van Laarhoven and E.H.L. Aarts, *Simulated Annealing: Theory and Applications*, Reidel Publishing, Dordrecht, Netherlands, 1987.

[23] J. Lam and J.-M. Delosme, "Performance of a New Annealing Schedule," *Proc. 25th ACM/IEEE DAC*, Anaheim, CA, 306-311, 1988.

[24] J. Lam and J.-M. Delosme, "Simulated Annealing: A Fast Heuristic for Some Generic Layout Problems," *Proc. IEEE ICCAD-88*, Santa Clara, CA, 510-513, 1988.

[25] S. Lin and B.W. Kernighan, "An Effective Heuristic Algorithm for the Traveling Salesman Problem," *Operations Research*, vol. 21, 498-516, 1973.

[26] M. Lundy and A. Mees, "Convergence of an Annealing Algorithm," *Math. Prog.*, vol. 34, 111-124, 1986.

[27] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, "Equation of State Calculations by Fast Computing Machines," *J. Chem. Phys.*, vol. 21, 1087-1092, 1953.

[28] R.H.J.M. Otten and L.P.P.P. van Ginneken, "Stop Criterion in Simulated Annealing," *Proc. IEEE ICCD*, Rye Brook, NY, 549-552, 1988.

[29] R.H.J.M. Otten and L.P.P.P. van Ginneken, *The Annealing Algorithm*, Kluwer Academic Publishers, Boston, MA, 1989.

[30] W. Padberg and G. Rinaldi, "Optimization of a 532-City Symmetric TSP," *Op. Res. Lett.*, vol. 6, 1-7, 1987.

[31] B. Preas, "Benchmarks for Cell-Based Layout Systems," *Proc. 24th ACM/IEEE DAC*, Miami Beach, FL, 319-320, 1987.

[32] F. Romeo and A. Sangiovanni-Vincentelli, "Probabilistic Hill Climbing Algorithms: Properties and Algorithms," *Proc. 1985 Chapel Hill Conf. on VLSI*, Chapel Hill, NC, 393-417, 1985.

[33] J.S. Rose, W.M. Snelgrove, and Z.G. Vranesic, "Parallel Standard Cell Placement Algorithms with Quality Equivalent to Simulated Annealing," *IEEE Trans. CADICS*, vol. 7, 387-396, 1988.

[34] J.S. Rose, W. Klebsch, and J. Wolf, "Temperature Measurement and Equilibrium Dynamics of Simulated Annealing Placements," *IEEE Trans. CADICS*, vol. 9, 253-259, 1990.

[35] C. Sechen and A. Sangiovanni-Vincentelli, "The TimberWolf Placement and Routing Package," *IEEE J. Solid-State Circuits*, vol. 20, 510-522, 1985.

[36] P. Sibani, J.M. Pedersen, K.H. Hoffmann, and P. Salamon, "Monte Carlo Dynamics of Optimization Problems: A Scaling Description," *Phys. Rev. A*, vol. 42, 7080-7086, 1990.

[37] P.N. Strenski and S. Kirkpatrick, "Analysis of Finite Length Annealing Schedules," *Algorithmica*, vol. 6, 346-366, 1991.

[38] S.R. White, "Concepts of Scale in Simulated Annealing," *Proc. IEEE ICCD*, Port Chester, NY, 646-651, 1984.

[39] D.F. Wong, H.W. Leong, and C.L. Liu, *Simulated Annealing for VLSI Design*, Kluwer Academic Publishers, Boston, MA, 1988.