A FAST METHOD FOR GENERALIZED STARTING TEMPERATURE DETERMINATION IN TWO-STAGE SIMULATED ANNEALING SYSTEMS

James M. Varanelli
James P. Cohoon

Computer Science Report No. CS-93-52
September 23, 1993

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

James M. Varanelli and James P. Cohoon
Department of Computer Science
University of Virginia
Charlottesville, VA 22903
USA

ABSTRACT

Simulated annealing is a stochastic process that has proven to be an effective method for approximating globally optimum solutions to many types of combinatorial optimization problems, especially in the field of VLSI computer-aided design. The major drawback to the simulated annealing algorithm is its typically very long running times. Several methods have been proposed for accelerating the simulated annealing algorithm. One method is to replace a significant portion of the stochastic operations with a fast heuristic. Simulated annealing can then begin from a lower starting temperature—a latter stage of the algorithm—to further improve the solution produced by the heuristic. This paper presents a method for approximating this starting temperature in general, as well as experience with two-stage systems for solving the VLSI partitioning, traveling salesperson, and minimum-length rectilinear Steiner tree problems.

1. INTRODUCTION

The simulated annealing process is an effective tool for computer-aided design (CAD) of VLSI circuits [1, 2, 8, 11, 15, 17, 19, 23, 25, 32]. This stems both from its applicability to a wide range of NP-hard combinatorial optimization problems and from the fact that it produces high quality approximate solutions to these problems. However, simulated annealing suffers from being very computationally expensive.

The simulated annealing process is used to search the *state space* of some particular combinatorial optimization problem, in order to find a configuration that closely approximates the global optimum of the given *cost function*, *c()*. This is accomplished by iteratively perturbing configurations according to some chosen *generation mechanism*. The generation mechanism will determine the size of the *neighborhoods*. A neighbohood is the set of all configurations reachable from the current configuration by one application of the generation mechanism. Acceptance of the perturbed configuration is determined by the *acceptance function*, which is a function of the *change in cost* of the perturbed configuration and its corresponding *temperature*. Change in cost, Δc, is equal to the difference in cost of the perturbed configuration and the configuration from which it was derived. Temperature, t, with respect to simulated annealing, refers to the stage at which the algorithm is currently executing. As the algorithm proceeds, the temperature is lowered from a predetermined initial value until the algorithm terminates at some temperature close to zero. In terms of implementation, the temperature is sometimes referred to as the *control parameter*. The manner in which the temperature is lowered is determined by the chosen *cooling schedule*. Cooling schedules will be discussed in more detail in Section 2.

The most common acceptance function used for simulated annealing implementations is known as the *Metropolis criterion* [24]. According to the Metropolis criterion, a perturbed configuration is accepted as the next configuration if $\Delta c \leq 0$. If instead $\Delta c > 0$, then the probability of the perturbed configuration being accepted is given by

$$e^{\left(\frac{-\Delta c}{t}\right)}$$

```
Simulated_Annealing()
{
  /*Get the initial state $i = i_0$ and the initial value of
    the control parameter $t = t_0$. Also set the cur-
    rent minimum-cost state equal to $i_0$.*/
  initialize($i$, $t$);
  $i_{min}$ = newminstate($i$);
  do {
    do {
      /*Create a new state $j$ by a small, random
        perturbation of state $i$.*/
      $j$ = perturb($i$);
      /*Calculate the difference in cost between
        state $j$ and state $i$.*/
      $\Delta c_{ij}$ = c($j$) - c($i$);
      /*Use Metropolis criterion to determine
        acceptance. random() is a pseudo-random
        number generator with a uniform distribu-
        tion over [0,1). If accepted, replace state $i$
        with state $j$.*/
      if (($\Delta c_{ij}$ <= 0) || (random() < exp(-$\Delta c_{ij}/t$)))
        $i$ = newstate($j$);
      /*If the cost of state $i$ is less than the cost of
        the current minimum-cost state, save state
        $i$ as the new minimum-cost state.*/
      if (c($i$) < c($i_{min}$))
        $i_{min}$ = newminstate($i$);
    } while ("not in equilibrium");
    /*Update the value of the control parameter.*/
    $t$ = update($t$);
  } while ("stop criterion has not been met");
  return($i_{min}$);
}
```

Figure 1: Pseudocode for the simulated annealing algorithm.

An outline of the simulated annealing algorithm using the Metropolis criterion is shown in Figure 1.

There has been considerable effort aimed at speeding up the simulated annealing algorithm. The majority of this work has concentrated on the development of faster cooling schedules [1, 7, 8, 11, 18, 19]. Another suggested approach is *two-stage simulated annealing* [9, 30, 31]. In a two-stage system, a fast heuristic is first used to replace the simulated annealing actions occurring at the highest temperatures of the cooling schedule, followed by a simulated annealing approach at the lower temperatures to further optimize the heuristic solution.

A major consideration for two-stage systems is the determination of the temperature at which to start the simulated annealing phase. If the starting temperature is too low, final solution quality will most likely be degraded. This arises from the fact that a significant amount of *probabilistic hill climbing* will most likely not be done due to the lower acceptance probabilities at the lower temperatures. Probabilistic hill climbing is a property of the algorithm that allows the acceptance of states with higher cost than the current state of the system according to a specific probabilistic acceptance criterion [29].

If instead, the starting temperature is too high, unnecessary work may be performed. In this case it is likely that too much probabilistic hill climbing will occur, essentially wasting some of the optimization resources used by the heuristic. This paper presents both a method for approximating the starting temperature and results from three two-stage simulated annealing systems that utilize the method to solve the VLSI partitioning, traveling salesperson, and minimum-length rectilinear Steiner tree problems [12, 14, 20].

Our proposed method for determining the starting temperature is a function of $c(i_{heur})$, the cost of the configuration returned by the first-stage heuristic; E_∞, the expected value of the cost function over the uniform distribution of configurations; and σ_∞^2, the variance of the cost function over the uniform distribution of configurations. Based on our experimental results, starting temperature can be approximated very closely at middle to low temperatures with the function

$$t(i_{heur}) = \frac{\sigma_{\infty}^{2}}{E_{\infty} - c(i_{heur})}$$

This method of approximation is based on results pertaining to the general behavior of the cost of the best-seen configuration over the course of the simulated annealing algorithm, $c(i_{min})$. In the above equation, i_{heur} is the configuration returned by the heuristic and $c(i_{heur})$ is its associated cost, which is assumed to be the cost of the best-seen configuration for a running simulated annealing process at the calculated starting temperature. Because our method is based on general behavior of the algorithm, it has the desirable property of being applicable to different problems as well as different simulated annealing formulations. Our method is described in more detail in Section 4.

Some effort has been previously directed at determining a starting temperature in two-stage simulated annealing systems [9, 30]. For these earlier approaches, a constant starting temperature was typically chosen based on experience with the problem as opposed to formalizing a method for true starting temperature determination. Unfortunately with such approaches, the previous constant starting temperature is of no value as soon as the simulated annealing formulation or the problem itself is changed. Only Rose, Klebsch, and Wolf present a generalized method of temperature determination [31]. This will be discussed in more detail in Section 3.

Section 2 gives some background on standard simulated annealing cooling schedules. Section 3 describes general two-stage simulated annealing systems and the problem of starting temperature determination. Section 4 presents our method for determining the starting temperature in two-stage systems. Sections 5, 6, and 7 present results from two-stage simulated annealing systems incorporating our method that are intended to solve the VLSI partitioning, traveling salesperson, and minimum-length rectilinear Steiner tree problems respectively.

2. COOLING SCHEDULES

Kirkpatrick, Gelatt, and Vecchi [15] first recognized that one can simulate the annealing process in order to generate sequences of configurations for the purpose of solving combinatorial optimization problems. Since their initial paper, many researchers have investigated the various aspects of the algorithm. One significant avenue of research has been on the cooling schedule [1, 11, 18, 19, 25, 29, 33]. There are four components to a cooling schedule.

- The initial value of the control parameter, t_0, corresponding to temperature;
- A rule for decrementing the value of the control parameter;
- A *stop criterion*, specifying conditions under which to terminate the algorithm; and
- A rule to determine the length of the sequence of moves at each value of the control parameter. The sequence is in fact a homogeneous Markov chain [29].

Cooling schedules tend to fall into one of two categories: *exponential* or *logarithmic* [2, 17]. Exponential schedules generally have a decrement rule that is independent of algorithmic evolution, usually of the form

$$t_n = t_0 \cdot \alpha^n$$

where α is some constant less than but close to one, usually in the range 0.90 - 0.99. A decrement rule of this type is often referred to as a *constant decrement rule*. Logarithmic schedules generally have decrement rules that vary dynamically over the course of the algorithm, usually of the form

$$t_n \sim t_{n-1} \cdot (\log \Gamma_{n-1})^{-1}$$

where Γ_{n-1} is some function of one or more statistics from the latest Markov chain, such as the mean or standard deviation of the cost [2, 17]. Since these statistics will change at each new temperature, decrement rules of this type are often called *variable decrement rules*. For each of the two-stage simulated annealing systems described later, two different cooling schedules are used in an effort to show that our method is compatible with schedules of each type as well as with different problems. The chosen exponential schedule is similar to the original one proposed by Kirkpatrick, Gelatt, and Vecchi [14]. We will refer to this as the *classic schedule*. The chosen logarithmic schedule was developed by Aarts and van Laarhoven [1].

For our implementations of these cooling schedules, the initial value of the control parameter is determined by a method independently described by Otten and van Ginneken [25] and White [33]. A large number of independent random configurations is generated. The scores for these configurations are then used to estimate the expected value, E_∞, and the variance, σ_∞^2, over the uniform distribution of configurations. We chose this number to be 10^3. Our experimental results indicate that this number is generally high enough to insure that the actual value of E_∞ lies within the 99% confidence interval of the computed value, regardless of the size of the state space under consideration. The value of t_0 can then be set such that

$$t_0 \geq \sigma_\infty$$

For all of our implementations, we chose to set $t_0 = \sigma_\infty$. It should be noted that the calculation of E_∞ and σ_∞^2 is central to our proposed method of starting temperature determination in two-stage annealing systems. For this reason, the above method for calculating t_0 is used in all of our implementations.

The classic schedule uses a constant decrement rule with either a constant or variable Markov chain length. For our implementations, we chose to set $\alpha = 0.95$ for the decrement rule. For our implementations employing a variable Markov chain length, the chains are terminated when either the number of accepted configurations equals the size of the neighborhoods or the total number of generated configurations equals twice the size of the neighborhoods, whichever comes first. The exact variation of the classic schedule used for each of the three problems examined later is described in the corresponding section.

The Aarts and van Laarhoven schedule uses a variable decrement rule with a constant Markov chain length. The decrement rule for the Aarts and van Laarhoven schedule is given by the equation

$$t_{k+1} = t_k \cdot \left(1 + \frac{t_k \cdot \ln(1+\delta)}{3\sigma_k}\right)^{-1}$$

where δ is a small positive constant called the *distance parameter*, and σ_k is the standard deviation of the cost for the Markov chain generated at temperature t_k [1]. For our implementations, we set $\delta = 0.085$ in order to achieve the desired quality of solution. The Markov chain length for the Aarts and van Laarhoven schedule is equal to the size of the neighborhoods.

Our implementations of the classic schedule terminate when four consecutive Markov chains end with the same value for the cost function. The Aarts and van Laarhoven schedule is terminated when the following relation holds:

$$\frac{\sigma_k^2}{t_k(E_\infty - E_k)} < \theta$$

where θ is a small positive constant called the *stop parameter* and E_k is the average value of the cost function for the Markov chain at temperature t_k [1]. For our implementations, we set $\theta = 0.00001$. Again, this value was chosen in order to insure the desired solution quality.

The various implementations of the simulated annealing algorithm described above [1, 15], as well as others not examined here [8, 11, 18, 19, 25], have proven to be quite effective for solving combinatorial optimization problems. With only a few exceptions, simulated annealing usually produces higher quality solutions than iterative improvement heuristics given the same amount of computation time [17]. Unfortunately, the algorithm suffers from usually prohibitive running times. Two-stage simulated annealing attempts to address this problem. The next two sections present an overview of two-stage annealing systems as well as a new method for starting temperature determination in two-stage systems.

3. CURRENT TWO-STAGE ANNEALING SYSTEMS

Two-stage simulated annealing addresses the method's problem of long computation times. Two-stage annealing systems consist of a heuristic algorithm designed to solve the given problem and a simulated annealing algorithm for the same problem. The heuristic algorithm is executed first. Simulated annealing is then performed in order to further improve the solution generated by the heuristic. Since a certain amount of optimization is performed by the chosen heuristic, simulated annealing can be started at a lower temperature than would normally be required to achieve the desired level of solution quality. If a heuristic is chosen that generates high quality solutions, a significant percentage of the simulated annealing algorithm can be skipped. This would correspond to the highest temperatures of the cooling schedule. If the chosen heuristic also has a low-order polynomial-bound complexity, time savings can be considerable. Thus, the major problem in the design of two-stage annealing systems is the determination of this starting temperature.

As noted above, most of the early work involving a two-stage simulated annealing approach made use of a constant starting temperature for the simulated annealing phase [9, 30]. This constant is generally derived through extensive experimentation with the chosen heuristic as well as the specific simulated annealing implementation being used. A constant starting temperature has the advantage of being computationally inexpensive once incorporated into the two-stage system. The obvious disadvanatge, however, is the fact that the constant is of no value once the heuristic, simulated annealing implementation, or the problem itself is changed.

Unfortunately, less effort has been directed at finding a general method for determining the starting temperature in two-stage simulated annealing systems. Rose, Klebsch, and Wolf [31] present a method for measuring the temperature of simulated annealing standard cell placements, which could be used in a two-stage standard cell placement system.

The Rose, Klebsch, and Wolf method involves approximating the probability distribution of the change in cost function by generating a large number of random moves from a given configuration that is assumed to be in quasi-equilibrium at some temperature, i.e. the configuration returned by the heuristic in a two-stage annealing system. This distribution is different at each temperature of a running simulated annealing process. The approximate distribution is used in a binary search-like algorithm to find the corresponding temperature. At each proposed temperature, the approximate distribution is used to calculate the magnitude of the expected value of negative cost moves and the magnitude of the expected value of positive cost moves. These values are compared and the next trial temperature is determined in a binary search-like manner. When these two values are found to be equal, the resulting temperature is returned as the quasi-equilibrium temperature of the given placement.

The above method produces good results when determining the temperature associated with a given solution to the standard cell placement problem. Unfortunately, the method has drawbacks. First, the accuracy of the method is both problem- and formulation-dependent. This arises from the fact that the approximation of the probability distribution for the change in cost function is measured from only one state [31]. The true distribution can only be found by measuring the distribution at all states. Thus, certain problems as well as certain formulations of the same problem will produce approximations of varying quality. In addition, the configuration used for the measurement process cannot be a local minimum. Since local minima have no lower cost neighbors, the measured distribution would be zero for all negative cost moves. This would generally not be the case in the true distribution [31]. As a result, the method cannot be used with greedy heuristics that use the same cost function as the simulated annealing formulation, since these formulations will usually terminate at a local minimum. This limits the number of applicable heuristics from which to choose. Finally, the method has a significantly long running time. This stems from the fact that a large number of random configurations (~10^5) must be generated to get an accurate probability distribution. This, in conjunction with the fact that a search procedure must be used in order to locate the corresponding temperature, leads to a computationally expensive method.

4. A NEW METHOD FOR STARTING TEMPERATURE DETERMINATION

Analysis of the Rose, Klebsch, and Wolf method does offer insight to desirable properties for a proposed method of starting temperature determination. First, the method should be generally applicable to all problems and all simulated annealing formulations. Second, the method should not be sensitive to the given starting configuration. Finally, the method should be as computationally inexpensive as possible. A method with these qualities is described here and is used in the two-stage simulated annealing systems described in the next three sections.

Our proposed method is based on the fact that any determined temperature will be an approximation to the actual temperature of the given configuration. The ideal approximation should be greater than the actual temperature to help insure that the proper amount of probabilistic hill climbing can occur while at the same time being as close as possible to the actual temperature in order to minimize the amount of unnecessary work done. Knowledge of the general behavior of the simulated annealing algorithm is essential to specifying a method of starting temperature determination that is to be generally applicable.

We first needed to determine what, if any, known general characteristics of the simulated annealing algorithm could be used as a basis for a method of starting temperature determination. Studies by Aarts and van Laarhoven [17] and Otten and van Ginneken [26] show the general behavior of the mean and standard deviation of the cost over the course of the algorithm. Unfortunately, the only information available prior to the simulated annealing phase in a two-stage system is the cost of the configuration returned by the heuristic. General

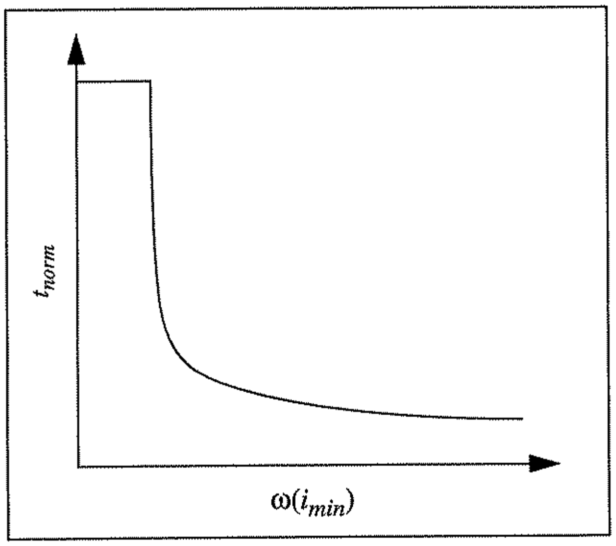
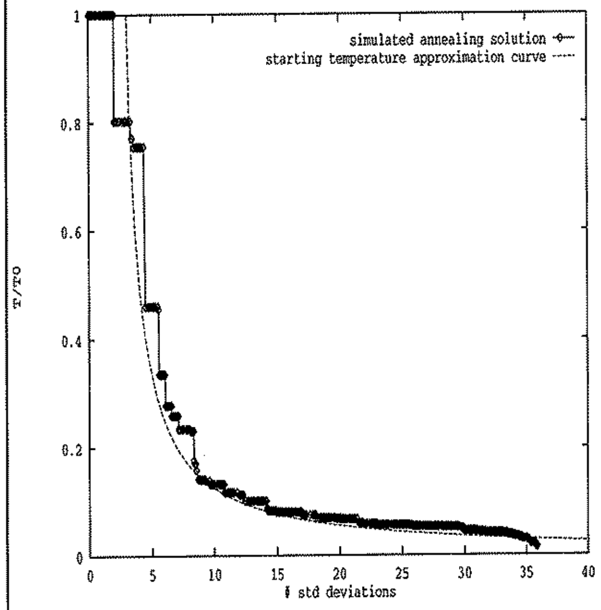


Figure 2: Behavior of minimum-cost configuration i_{min} with respect to distance ω from E_∞ vs. normalized temperature t_{norm}.

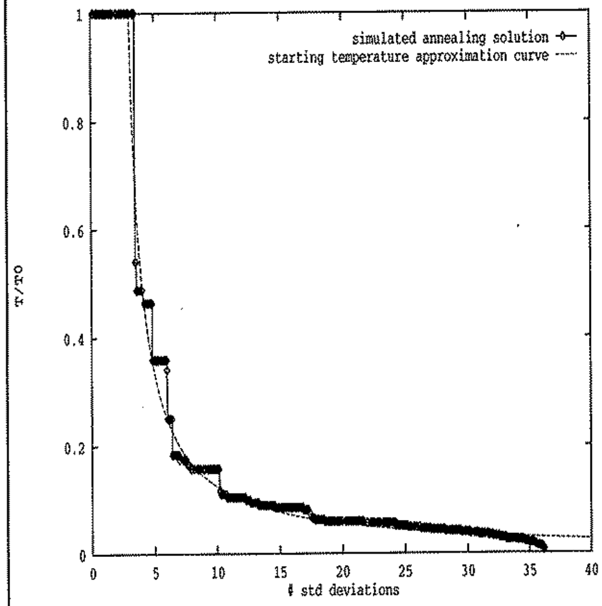
behavior of the mean or standard deviation of the cost is of little value since neither can be extrapolated from the cost of a single configuration without a method similar to that of Rose, Klebsch, and Wolf. If, however, we assume the heuristic solution to be the current simulated annealing minimum-cost configuration, knowledge of the behavior of the minimum-cost configuration over the course of the algorithm could be used in a general method for starting temperature determination.

We examined the behavior of the minimum-cost configuration, i_{min}, over a large number of runs of the simulated annealing algorithm. Both the actual score of the minimum-cost configuration and its corresponding temperature were found to be too problem- and formulation-dependent to give a true general behavior at decreasing temperatures. A more general way to express the score of the minimum-cost configuration seen so far, $c(i_{min})$, is its distance in standard deviation units from the expected value over the uniform distribution of states E_∞. This distance, ω, is easily calculated for any configuration i with the function

$$\omega(i) = \frac{E_\infty - c(i)}{\sigma_\infty}$$

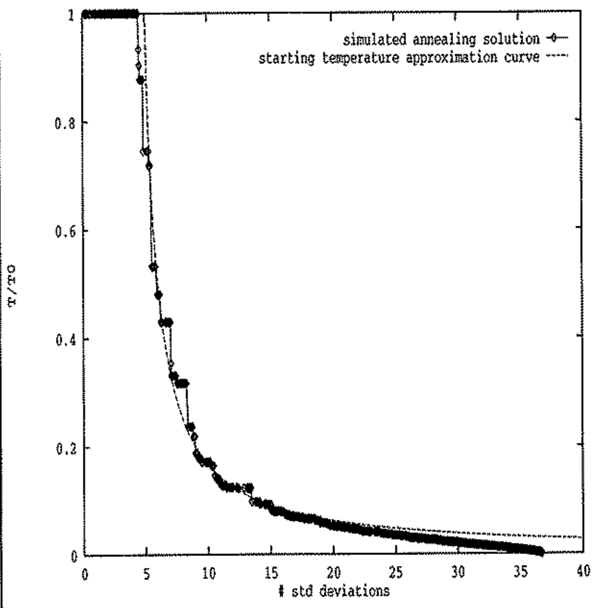


(a)

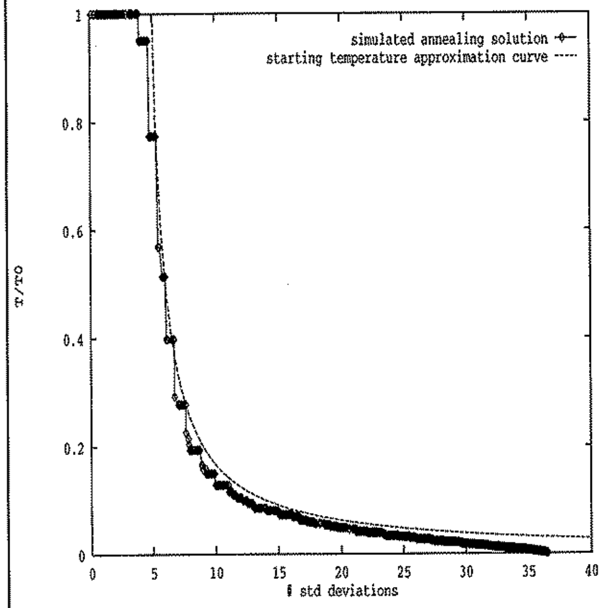


(b)

Figure 3: Approximation curve compared to simulated annealing solutions to the bipartitioning of SIGDA Benchmark Primary1. Plots (a) and (b) concern respectively the schedules of Aarts and van Laarhoven and Kirkpatrick, Gelatt, and Vecchi.



(a)



(b)

Figure 4: Approximation curve compared to simulated annealing solutions to the 318 city TSP instance of Lin and Kernighan. Plots (a) and (b) concern respectively the schedules of Aarts and van Laarhoven and Kirkpatrick, Gelatt, and Vecchi.

where $c(i)$ is the cost of configuration i. This value plotted against a normalized temperature $t_{norm} = t_k/t_0$, gives a good indication of the general behavior of the minimum-cost configuration over the course of the algorithm. A generic plot of this type is shown in Figure 2.

After many runs using both of the cooling schedules described in Section 2 for our three chosen problems, it became clear that the normalized temperature is inversely proportional to the distance $\omega(i_{min})$. More precisely, the normalized temperature can be approximated quite closely by

$$t_{norm}(i_{min}) = \frac{\sigma_\infty}{E_\infty - c(i_{min})}$$

Figures 3 and 4 illustrate this relationship. For these figures, the value of c_{min} in terms of distance ω from E_∞ was tracked over the course of actual simulated annealing runs and plotted against normalized temperature. These resulting curves are then compared with the curve defined by the above equation.

Using this fact, the actual starting temperature approximation t_{approx} for a two-stage simulated annealing system can be found by:

$$t_{approx}(i_{heur}) = \frac{t_0\sigma_\infty}{E_\infty - c(i_{heur})}$$

where c_{heur} is the cost of the configuration returned by the heuristic and is assumed to be the current minimum value of the cost function for the simulated annealing phase (i.e. c_{heur} is an approximation of c_{min}). If t_0 is set equal to σ_∞ as described in Section 2, then the numerator is simply the variance over the uniform distribution of configurations and the formula becomes:

$$t_{approx}(i_{heur}) = \frac{\sigma_\infty^2}{E_\infty - c(i_{heur})}$$

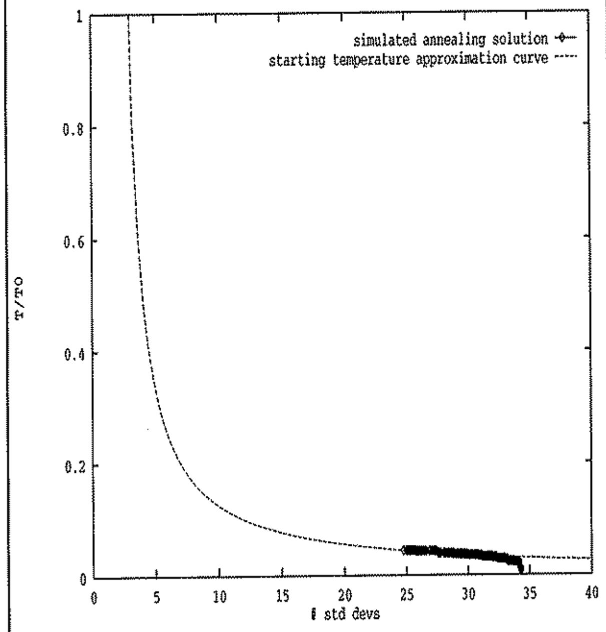
The above observations form the basis of our method for approximating the true starting temperature in a two-stage simulated annealing system. Our method can be summarized with the following algorithm:

- Execute the problem-dependent heuristic to obtain the initial configuration and its corresponding cost c_{heur}
- Obtain estimates for expected value E_∞ and variance $\sigma_\infty{}^2$ of the cost over the uniform distribution of configurations using the technique described in Section 2.
- Use c_{heur}, E_∞, and $\sigma_\infty{}^2$ in the above formula to obtain the starting temperature approximation t_{approx}.
- Set $t = t_{approx}$ and begin the simulated annealing phase.

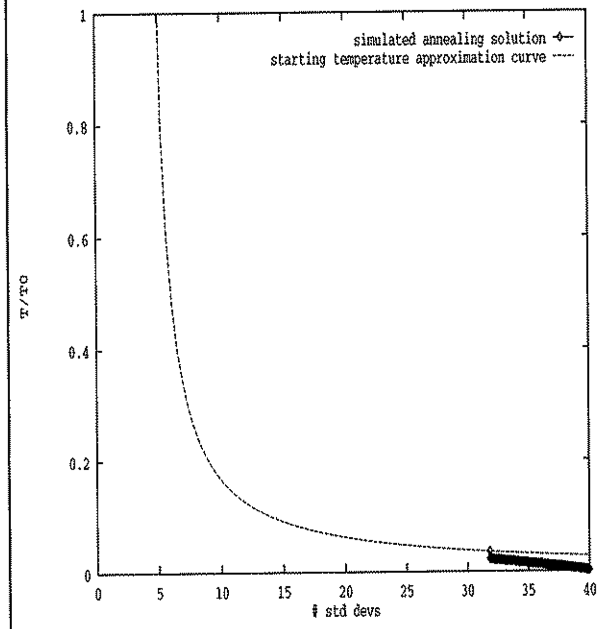
Figures 3 and 4 indicate that our method does indeed produce approximations that are quite close to the true temperatures at which the configuration in question would be produced during an actual simulated annealing run. Although we only show plots for two of the three problems we are considering, it should be noted that the corresponding plots for the minimum-length rectilinear Steiner tree problem show similar behavior. Our approximation curves show close agreement with actual simulated annealing runs independent of the problem being considered or the cooling schedule being used. Figure 5 indicates that once the simulated annealing phase begins from the determined starting temperature, the algorithm has the expected convergence behavior. Again, while we plot this for only two of our chosen problems and one schedule type per problem, the corresponding plots for the other schedule-problem combinations are similar. These plots indicate that there is significant further optimization being performed during the simulated annealing phase to improve upon the solution obtained by the heuristic. The next three sections present results for a variety of problems from two-stage annealing systems incorporating our method of starting temperature determination.

5. THE VLSI PARTITIONING PROBLEM

The input to the VLSI partitioning problem consists of a set of circuit elements, or *cells*, connected



(a)



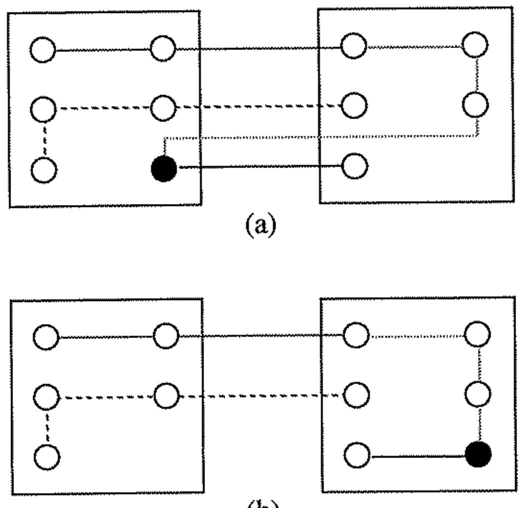
(b)

Figure 5: Approximation curve vs. post-heuristic simulated annealing solution. (a) Primary1 instance using the Aarts and van Laarhoven schedule. (b) 318 city TSP instance using the Kirkpatrick, Gelatt, and Vecchi schedule.

by a set of *nets*. A net electrically connects a group of at least two cells. *Pins* are the interconnection points on cells. The goal of the VLSI partitioning problem is to partition the cells into two blocks so as to minimize the number of nets that have cells in both blocks. This problem is often referred to as the *mincut partitioning problem*, since the goal is to minimize the number of nets that are cut by the partition. Figure 6 illustrates the VLSI partitioning problem.

There is often a balance criterion associated with block assignment. A value is given that specifies the maximum percentage of the sum of the sizes of the cells that may be in one block. This prevents the migration of all blocks onto the same side of the partition. Usually this value is given as 50%. There is also a tolerance associated with the balance criterion. This tolerance is normally chosen to be the size of the largest cell [6].

In order to use a simulated annealing approach,



(b)

Figure 6: The VLSI partitioning problem. The above example has 11 cells (circles) connected by four nets (hashed lines). (a) All four nets are cut by the partition. When the filled cell is moved, (b), two nets are no longer cut.

configurations must be defined and a generation mechanism must be specified. A configuration for the VLSI partitioning problem is simply the specification of the block in which each cell is currently placed. A new configuration is generated from the current configuration by changing the block of a randomly chosen cell as long as the new configuration meets the balance criterion. If the chosen cell would cause imbalance by changing its partition, another cell is chosen at random until one is found that meets the balance criterion if moved. This generation mechanism produces a neighborhood structure of size equal to the total number of cells.

Each configuration specifies the number of nets that contain cells on both sides of the partition. This is the number to be minimized and is the basis for a simulated annealing cost function. An additional term is added to the cost function to take into account the balance criterion. If the balance criterion is specified as 50%, the cost of a configuration i is given by

$$c(i) = |E_{cut}| + \lambda \cdot (|A| - |B|)^2$$

where $|E_{cut}|$ is the number of nets cut by the partition, $|A|$ and $|B|$ are the sums of the sizes of the cells in each of the two blocks of the partition respectively, and λ is an imbalance factor. We chose to let $\lambda = 0.02$ according to experimental results presented by Lam and Delosme [19].

The Fiduccia and Mattheyses heuristic [6] was selected for use as the first stage in the two-stage simulated annealing system. It is closely related to the method of Kernighan and Lin [14]. This heuristic was selected due to its fast running times and quality of solution. The complexity of the algorithm is shown to be linear in the total number of pins. Fiduccia and Mattheyses point out that their algorithm generally converges to a final solution in a small number of passes, with the bulk of the optimization being done in the first pass.

Our own experimental results show that one pass of the Fiduccia-Mattheyses algorithm gives solutions that on average are 5-25% less than simulated annealing solutions in terms of ω, the distance from E_∞ measured in standard deviation units, with the majority of the solutions closer to the lower end of the range. In order to minimize the time used by the first stage, only one pass of the Fiduccia-Mattheyses algorithm is used as a precursor to simulated annealing in the two-stage VLSI partitioning system. .

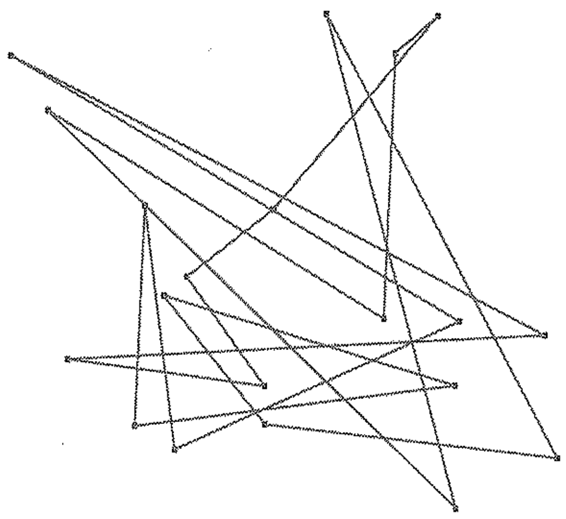
Data instance (cells)	SA CPU time (sec)	SA tour length	Two-stage CPU time (sec)	Two-stage tour length	% CPU time decrease
50	1.26	30.7	1.01	30.9	19.8
100	4.89	63.7	2.49	63.7	49.1
250	33.13	154.5	14.44	156.2	56.4
500	168.44	306.0	61.52	308.4	63.5
1500	2286.42	904.6	712.75	907.5	68.8
Primary1	618.20	81.2	187.93	82.0	69.6

Table 1: Results for a two-stage VLSI partitioning system using the Aarts and van Laarhoven schedule

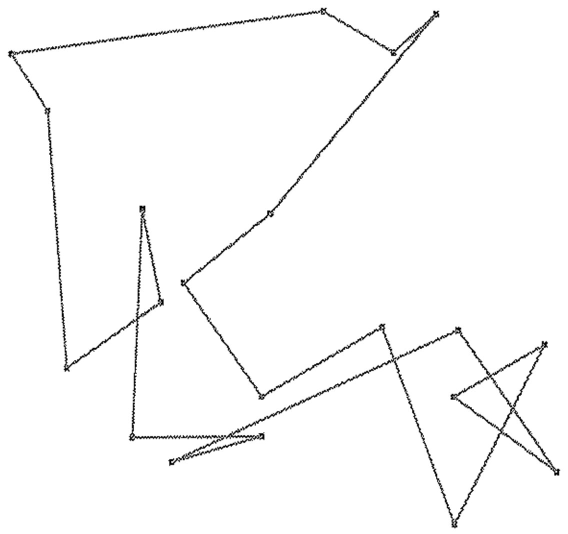
Data instance (cells)	SA CPU time (sec)	SA tour length	Two-stage CPU time (sec)	Two-stage tour length	% CPU time decrease
50	1.44	30.1	1.21	30.3	16.0
100	5.26	62.8	3.42	62.6	35.0
250	24.68	154.2	15.82	157.5	35.9
500	82.14	310.9	56.12	309.5	31.7
1500	724.38	926.6	461.01	928.9	36.4
Primary1	261.12	108.4	160.43	107.5	38.6

Table 2: Results for a two-stage VLSI partitioning system using a variation of the Kirkpatrick, Gelatt, and Vecchi schedule.

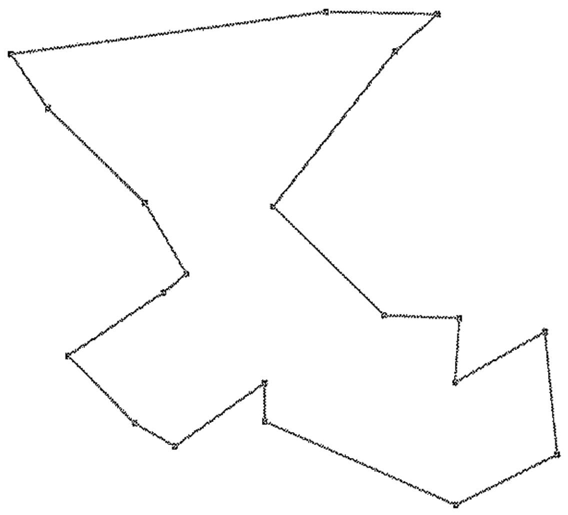
As noted in Section 2, both the classic cooling schedule and the schedule of Aarts and van Laarhoven are used for the simulated annealing phase. Our implementation of the classic schedule features a constant decrement rule with $\alpha = 0.95$ and a variable Markov chain length. Markov chains are terminated after the number of accepted configurations equals the size of the neighborhoods or the



(a)



(b)



(c)

Figure 7: Evolution of the best-seen solution for a randomly generated 20 city TSP using a two-stage SA system. (a) Random initial solution with cost = 751.13 and temperature = 60.90. (b) Solution after first-stage heuristic with cost = 364.78 and temperature = 12.98. (c) Final solution after SA phase with cost = 262.09 and temperature = 0.72.

total number of generations equals two times the size of the neighborhoods. The algorithm is stopped when four consecutive Markov chains end with the same value for the cost function.

All algorithms discussed in this and other sections are implemented in the C programming language and executed on a Sun SparcStation 2. Experimental data used for evaluating the two-stage VLSI partitioning system is made up of twenty-five randomly generated networks with average edge degrees generally greater than five, plus the SIGDA standard cell benchmark circuit Primary1 [28]. The network sizes range from 50 cells with 50 nets to 1500 cells with 1500 nets. For the twenty-five randomly generated networks, there are five instances for each of five different-sized networks. The results discussed for each network size are the average results of the five random instances of the network size in question. Each random instance was averaged over two runs, for a total of ten runs for each network size. The results for Primary1 are also averaged over ten runs. Results are gathered for both cooling schedules. The results are given in Tables 1 and 2 for the Aarts and van Laarhoven schedule and the classic schedule respectively. As can be seen from the tables, significant speedup is observed in the two-stage systems over standard simulated annealing, while average solution quality for the two-stage systems deviates from standard simulated annealing solution quality by only 0.7% and 0.2% for the Aarts and van Laarhoven schedule and the classic schedule respectively.

6. TRAVELING SALESPERSON PROBLEM

The input to the traveling salesperson problem consists of a symmetric $n \times n$ distance matrix d, representing distances between n cities [20, 27]. The goal is to find a minimum-length tour that visits each city exactly once while terminating at the city of origin.

A configuration is simply a list of cities, each appearing exactly once and in the order specified by the current tour. A common generation mechanism for a simulated annealing approach to the traveling salesperson problem is the *inversion* or *2-opt transition* first introduced by Croes [4] and later by Lin [21]. A 2-opt transition consists of choosing two cities at random from the current tour and reversing the order of the cities between them. This generation mechanism defines a neighborhood structure of size $n(n-1)/2$. The cost of a configuration is the sum of the distances between the cities specified by the current tour. More precisely, the cost function for the traveling salesperson problem is:

$$c(\pi) = \sum_{i=1}^{n-1} d_{\pi_i \pi_{i+1}} + d_{\pi_n \pi_1}$$

where π is a permutation of the list of cities ordered according to the current tour.

A natural choice for the heuristic phase of the two-stage annealing system is an algorithm based on the 2-opt heuristic presented by Croes [4]. Our experimental results show that solutions produced by our variation of the Croes algorithm are on average 5-20% less than simulated annealing solutions in terms of ω, the distance from E_∞ measured in standard deviation units. The Croes algorithm generally produces relatively better-quality solutions to the traveling salesperson problem than does one pass of the Fiduccia-Mattheyses algorithm for the VLSI partitioning problem. However, the Croes algorithm is more computationally expensive than the Fiduccia-Mattheyses algorithm. Figure 7 illustrates the evolution of the best-seen solution for the two-stage traveling salesperson system..

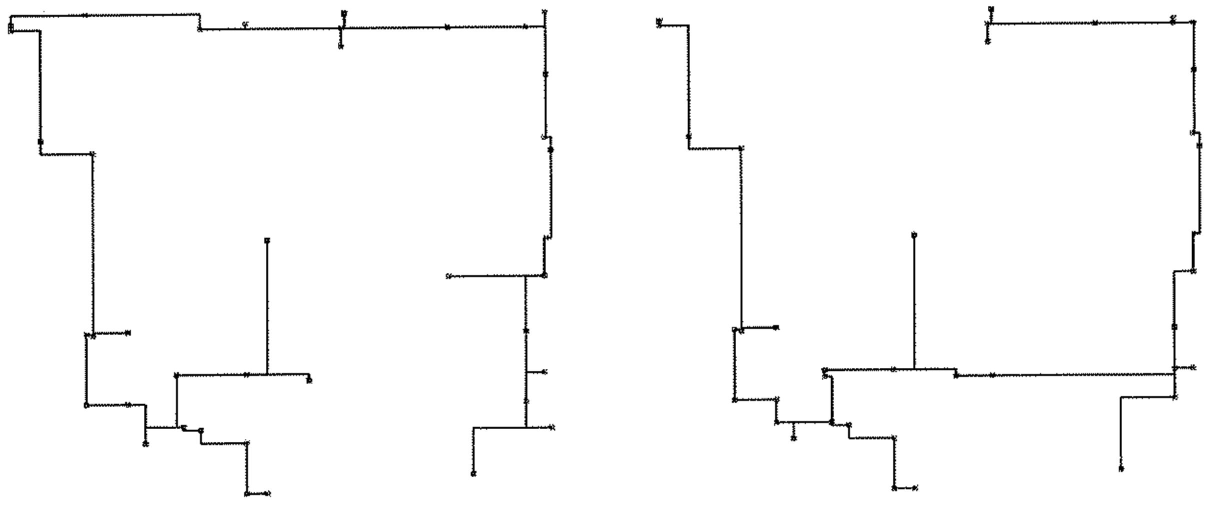
Data instance (cities)	SA CPU time (sec)	SA tour length	Two-stage CPU time (sec)	Two-stage tour length	% CPU time decrease
20	2.21	258.6	1.38	254.3	37.6
42	23.43	704.6	12.43	703.7	46.9
50	36.37	240.3	17.66	236.4	51.4
57	52.55	13075.3	24.31	13133.0	53.7
100	309.51	316.0	139.65	307.3	54.9
318	13633.97	42943.7	5588.25	42835.0	59.0

Table 3: Results for a two-stage traveling salesperson system using the Aarts and van Laarhoven schedule

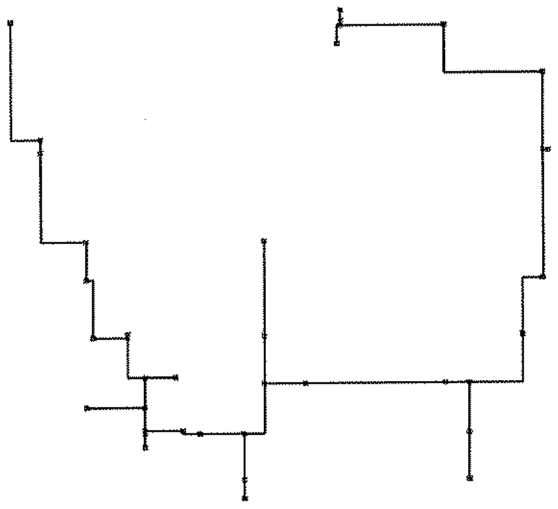
Data instance (cities)	SA CPU time (sec)	SA tour length	Two-stage CPU time (sec)	Two-stage tour length	% CPU time decrease
20	0.90	256.6	0.67	255.1	25.6
42	5.31	705.4	3.41	704.8	35.8
50	7.23	236.3	4.39	238.0	39.7
57	11.61	13164.6	6.53	13106.5	43.8
100	42.38	326.8	21.03	320.8	50.4
318	1088.81	43347.7	363.62	43360.8	66.6

Table 4: Results for a two-stage traveling salesperson system using a variation of the Kirkpatrick, Gelatt, and Vecchi schedule.

Two different cooling schedules are again used for the simulated annealing phase. The first schedule is again that of Aarts and van Laarhoven. The second schedule is another variation on the classic schedule. A constant decrement rule with $\alpha = 0.95$ and a constant Markov chain length equal to the size of the neighborhoods are used in this particular



(a) (b)



(c)

Figure 8: Evolution of the best-seen solution for a randomly generated 20 terminal network in a two-stage SA system intended to solve the RSMT problem. (a) Random initial solution with cost = 4512 and temperature = 225.76. (b) Solution after first-stage heuristic with cost = 3929 and temperature = 60.74. (c) Final solution after SA phase with cost = 3584 and temperature = 0.91.

implementation of the classic schedule. As before, the algorithm is terminated when four consecutive Markov chains end with the same value for the cost function.

Experimental data used for evaluating the two-stage traveling salesperson system consists of the following: the 20 city problem of Croes [4]; the 42 city problem of Dantzig, Fulkerson, and Johnson [5]; a randomly generated 50 city problem; the 57 city problem of Karg and Thompson [13]; a randomly generated 100 city problem; and the 318 city problem of Lin and Kernighan [22]. Results are again gathered for both cooling schedules, taking the average of ten runs for each problem. The results are given in Tables 3 and 4. As is the case with the two-stage VLSI partitioning system, significant speedup is noted over standard simulated annealing. Average solution quality for the two-stage systems is slightly better than that of standard simulated annealing by 1.2% for the Aarts and van Laarhoven schedule and 0.4% for the classic schedule.

7. MINIMUM-LENGTH RECTILINEAR STEINER TREE PROBLEM

The input to the minmum-length rectilinear Steiner tree (RSMT) problem consists of a set of n points in a plane, called *terminals* [12]. 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*, may also be used to connect the terminals. The possible locations of the Steiner points lie on a grid defined by the locations of the terminals. Using a result of Hanan [10], we can restrict the locations of the Steiner points to 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 $n - 1$.

A configuration is a specification of the interconnections among the n terminals and the chosen $n - 1$ Steiner points. Our generation mechanism consists of randomly selecting one of the Steiner points and moving it to a randomly selected currently unused Steiner location. This generation mechanism defines a neighborhood structure of size equal to the number of possible Steiner locations. The cost of a configuration is the sum of the lengths of the line segments connecting the terminals and the Steiner points. More explicitly,

$$c(i) = \sum_{l \in L} length(l)$$

where L is the set of all line segments used in the construction of configuration i and *length(l)* is the manhattan distance for the two endpoints that define the edge $l \in L$. .

Data instance (terminals)	SA CPU time (sec)	SA tree length	Two-stage CPU time (sec)	Two-stage tree length	% CPU time decrease
9	20.19	1554.4	16.06	1554.4	20.5
11	82.21	2822.0	51.94	2822.0	36.8
13	47.87	1950.6	18.20	1949.6	62.0
16	198.90	3008.4	81.55	3002.4	59.0
20	3287.86	304.9	2284.25	304.5	30.5
30	21654.18	359.3	15502.77	359.3	28.4

Table 5: Results for a two-stage RSMT system using the Aarts and van Laarhoven schedule.

Data instance (terminals)	SA CPU time (sec)	SA tree length	Two-stage CPU time (sec)	Two-stage tree length	% CPU time decrease
9	21.83	1554.2	17.33	1554.0	20.6
11	55.90	2822.0	38.63	2822.0	30.9
13	41.63	1946.8	27.83	1942.6	33.1
16	132.11	2999.6	85.82	2998.8	35.0
20	1057.09	305.4	738.32	304.7	30.2
30	5360.77	360.2	3772.57	359.9	31.8

Table 6: Results for a two-stage RSMT system using a variation of the Kirkpatrick, Gelatt, and Vecchi schedule.

The heuristic chosen for the first phase of the two-stage RSMT system is based on Kruskal's minimum-spanning tree algorithm [16]. Kruskal's algorithm is first run to obtain the minimum-spanning tree for the n terminals using no Steiner points. The $n - 1$ Steiner points are then added in a greedy fashion from the set of possible Steiner locations to form the initial configuration for the simulated annealing phase. Solutions produced by our variation of Kruskal's algorithm are on average 5-20% less than simulated annealing solutions in terms of ω, the distance from E_∞ measured in standard deviation units. Figure 8 illustrates the evolution of the best-seen solution for the two-stage RSMT system.

As is the case for the two previous problems, both the Aarts and van Laarhoven schedule and a variation of the classic schedule are used for the simulated annealing phase. The implementation of the classic schedule includes a constant decrement rule with $\alpha = 0.95$ and a constant Markov chain length equal to the size of the neighborhoods. As is the case with the two previous implementations of the classic schedule, the algorithm terminates when four consecutive Markov chains end with the same value for the cost function.

Experimental data used for evaluating the two-stage RSMT system consists of four of the larger nets from the SIGDA benchmark Primary1[28] as well as randomly generated 20 and 30 terminal networks The placements of the nets taken from Primary1 are intermediate solutions generated by the Sharp placement and routing package [3]. The nets taken from Primary1 range in size from 9 terminals to 16 terminals. Results are again gathered for both cooling schedules, taking the average over ten runs for each network. The results are shown in Tables 5 and 6. As is the case with the previous two problems, significant speedup is noted for the two-stage RSMT system over standard simulated annealing. Average solution quality for the two-stage systems is again slightly better than that of standard simulated annealing by 0.1% for both the Aarts and van Laarhoven schedule and the classic schedule.

8. CONCLUSION

A method is presented to determine the starting temperature of the simulated annealing phase in two-stage simulated annealing systems. The method is experimentally shown to be generally applicable to different problems and formulations. The resulting two-stage systems have significantly lower running times than simulated annealing alone with no loss in solution quality.

9. ACKNOWLEDGEMENTS

The authors' work have been supported in part through National Science Foundation grants MIP-9107717 and CDA-8922545 and Virginia CIT award 5-30971. Their support is greatly appreciated.

10. 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] R. Azencott, Ed., *Simulated Annealing: Parallelization Techniques*, John Wiley and Sons, New York, NY, 1992.

[3] S. Bapat and J.P. Cohoon, "Sharp-Looking Partitioning," *Proc. 2nd IEEE EDAC,* Amsterdam, Netherlands, 172-176, 1991.

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

[5] 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.

[6] 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, 1985.

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

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

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

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

[11] 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.

[12] F.K. Hwang, D.S. Richards, and P. Winter, *The Steiner Tree Problem*, North-Holland, Amsterdam, Netherlands, 1992.

[13] R.L. Karg and G.L. Thompson, "A Heuristic

Approach to Solving Traveling-Salesman Problems," *Management Science*, vol. 10, 225-247, 1964.

[14] B.W. Kernighan and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs," *Bell System Tech. J.*, vol. 49, 291-307, 1970.

[15] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, 45-54, 1983.

[16] 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.

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

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

[19] 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.

[20] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys, Ed., *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, John Wiley and Sons, New York, NY, 1985.

[21] S. Lin, "Computer Solutions of the Traveling Salesman Problem," *Bell System Tech. J.*, vol. 44, 2245-2269, 1965.

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

[23] M. Lundy, "Applications of the Annealing Algorithm to Combinatorial Problems in Statistics," *Biometrika*, vol. 72, 191-198, 1985.

[24] 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.

[25] R.H.J.M. Otten and L.P.P.P. van Ginneken, "Annealing Applied to Floorplan Design in a Layout Compiler," *Proc. Automation '86*, Houston, TX, 185-228, 1986.

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

[27] C.H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, New York, NY, 1982.

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

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

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

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

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

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