A Two-Stage Simulated Annealing Methodology

James M. Varanelli and James P. Cohoon†

Department of Computer Science
University of Virginia
Charlottesville, VA 22903

ABSTRACT

We propose a two-stage simulated annealing method. While most previous work has focused on ad hoc constant starting temperatures for the low temperature annealing phase, this paper presents a more formal method for starting temperature determination in two-stage simulated annealing systems. We have successfully applied our method to three optimization problems using both classic and adaptive schedules. We also briefly discuss an alternative stop criterion that experimentally reduces the running time up to an additional ten percent in our problem suite.

1. INTRODUCTION

The simulated annealing (SA) algorithm [4] is an effective tool in the field of VLSI computer-aided design. This stems from both its general applicability to a wide range of combinatorial optimization problems and that it consistently produces high quality solutions for these problems. SA has only one significant disadvantage—its typically very long running time.

There has been considerable effort aimed at speeding up traditional monotone cooling SA. One such approach is two-stage simulated annealing (TSSA) [8]. For TSSA, a faster heuristic is used to replace the SA actions occurring at the highest temperatures in the cooling schedule. Conventional SA is then initiated at a low temperature in an attempt to improve the heuristic solution.

The principal consideration in the design of a TSSA system is determining the starting temperature for the SA phase. Earlier approaches are based on finding a reasonable constant starting temperature. The obvious disadvantage of this method lies in the fact 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 [8] present a more formal method for determining the starting temperature. Their method is based on Markov equilibrium dynamics and it produces good results for the standard cell placement problem. However, there are problem- and formulation-dependent constraints on the choice of first-stage heuristic and a high computational cost that have discouraged widespread adoption.

This paper presents a new method for generalized starting temperature determination in traditional TSSA systems. Background information used in the derivation of the method is given in Section 2; Section 3 presents the new method; Section 4 gives our experimental results for three different combinatorial optimization problems, namely the VLSI network partitioning (VLSI-NPP), rectilinear Steiner minimal tree (RSMT), and traveling salesman problem (TSP) problems; and Section 5 describes our current work with an alternative stop criterion.

2. BACKGROUND INFORMATION

One way of ensuring general applicability of the methodology is to base the determination of the starting temperature on some characteristic behavior of the traditional SA algorithm. We choose the behavior of the best-so-far (BSF) solution $i_{BSF}$ to serve as the basis for our method. The reason for this choice lies in the fact that the cost of the solution returned by the heuristic, i.e., the BSF solution, is the only piece of information available at the beginning of the SA phase in a TSSA system.

Since absolute cost is problem-dependent, a normalization scheme is required for consistency across different problems. Let $E_\infty$ and $\sigma_\infty$ respectively represent the expected cost and the standard deviation of the cost over all solutions in the state space. We can normalize the cost $c_{norm}(i)$ of a solution $i$ by measuring it in standard deviation units $\sigma_\infty$ away from the expected cost $E_\infty$. We can normalize the temperature $t_{norm}$ by dividing it with respect to $T_0$, the initial temperature in a SA algorithm. Explicitly, this leads to the two normalizations:

$$c_{norm}(i) = (E_\infty - c(i)) / \sigma_\infty$$

and

†The authors gratefully acknowledge the support of the National Science Foundation through grants MIP-9107717 and CDA-8922545.

1066-1395/95 $4.00 © 1995 IEEE
Figure 1: SA solution curve $t_{\text{norm}}$ vs. $c_{\text{norm}}(i_{\text{BSF}})$.

$$t_{\text{norm}} = t_k/t_0$$  \hspace{1cm} (2)

where $c(i)$ is the absolute cost of the current solution $i$ and $t_k$ is the $k^{th}$ SA temperature value.

Numerical studies have been conducted for different large-scale pseudo-random combinatorial problems examining solution densities at varying SA temperatures by four different sets of authors [3, 6, 7, 10]. All four independently present evidence that supports a typical behavior of the expected cost $E_k$ and standard deviation $\sigma_k$ with respect to SA temperature $t_k$. 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:

$$E_k = E_{\infty} - \left( \sigma_{\infty}^2/t_k \right)$$  \hspace{1cm} (3)

and

$$\sigma_k = \sigma_{\infty}$$  \hspace{1cm} (4)

Additionally, the investigations independently show that the probability distribution of the cost values generated during the $k^{th}$ Markov chain can be closely approximated by a normal distribution. These preliminaries are used in the derivation of the proposed method for starting temperature determination presented in the next section.

3. DERIVATIONS

Given the background information in the previous section, we present the following two propositions that describe the derivation of our 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 solution $i$ would be found as the BSF solution:

$$t_k(i) = \frac{\sigma_{\infty}^2}{E_{\infty} - c(i) - \gamma_{\infty} \sigma_{\infty}}$$  \hspace{1cm} (5)

Proof: See [9].

Equation 5 serves as the basis for our proposed method of starting temperature determination. However, the offset $\gamma_{\infty}$ is still an unknown. The offset represents the difference between the expected cost $E_k$ and the BSF cost $c(i_{\text{BSF}})$ at SA temperature $t_k$. It is important to note that without the offset term the temperature approximation is too low, resulting in TSSA solution quality less than that of standard SA. Proposition 2 describes the calculation of $\gamma_{\infty}$.

Proposition 2: Given a SA formulation for some combinatorial optimization problem with Markov chain length $L_M$, the offset $\gamma_{\infty}$ can be calculated probabilistically with the equation:

$$P \left[ E_{\infty} - \gamma_{\infty} \sigma_{\infty} < X < E_{\infty} + \gamma_{\infty} \sigma_{\infty} \right] = 1 - \left| L_M \right|^{-1}$$  \hspace{1cm} (6)

Proof: See [9].

Using Equation 6, the value for $\gamma_{\infty}$ can be calculated via numerical methods or by table lookup. Table 1 shows results for approximating the offset in a suite of VLSI-NPP instances using table values. Results for each instance are averaged over 20 runs. As can be seen in the table, computed $\gamma_{\infty}$ values match very closely with observed offsets. Results are similar for the other two problems in our test suite [9].

Based on the above discussion, our method can be summarized in the following steps:

- Execute the heuristic to obtain $c(i_{\text{BSF}})$.
- Obtain values for $E_{\infty}, \sigma_{\infty}$, and $\gamma_{\infty}$.
- Use $c(i_{\text{BSF}}), E_{\infty}, \sigma_{\infty}$, and $\gamma_{\infty}$ in Equation 5 to get $t_{\text{app}}$.
- Set $t = t_{\text{app}}$ and begin the SA phase.

| cells  | $L_M$ | $1-|L_M|^{-1}$ | $\gamma_{\infty}$ computed | $\gamma_{\infty}$ observed |
|--------|-------|----------------|---------------------------|---------------------------|
| 100    | 100   | 0.9900         | 2.58                      | 2.76                      |
| 500    | 500   | 0.9980         | 3.09                      | 3.02                      |
| 833    | 833   | 0.9988         | 3.22                      | 3.11                      |
| 1500   | 1500  | 0.9993         | 3.36                      | 3.18                      |
| 3014   | 3014  | 0.9995         | 3.52                      | 3.34                      |
| 10000  | 10000 | 0.9998         | 3.86                      | 3.88                      |

Table 1: Experimental results for approximating the offset for several VLSI-NPP instances.
As can be seen in Figure 1, our method produces approximations that are quite close to actual SA temperatures associated with the BSF solution. The experimental results presented in the next section indicate that there is a significant time reduction seen for TSSA systems incorporating the above methodology over standard SA with no loss in solution quality.

4. TSSA EXPERIMENTAL RESULTS

Results are presented for the three aforementioned optimization problems. Each problem in our test suite is solved with two different cooling schedules—the classic schedule [4] and an adaptive schedule [6]. All TSSA systems are implemented in the C/C++ programming language and executed on a Sun SparcServer 10/51^TM^. All results are averaged over 10 runs.

The TSSA VLSI-NPP system incorporates the Fiduccia and Mattheyses (F-M) heuristic [2]. We use only one pass of the algorithm, since the majority of improvement takes place during the first pass. Experimental data used for evaluating the TSSA VLSI-NPP system is made up of SIGDA standard cell benchmark circuits PrimarySC1 (833 cells, 904 nets) and PrimarySC2 (3014 cells, 3029 nets) as well as four randomly generated networks with average edge degrees and net distributions similar to the benchmark circuits. The network sizes ranged from 100 nets with 100 cells to 10000 nets with 10000 cells. The results are given in Tables 2 and 3 respectively for the adaptive and classic schedules. The average speedup over SA is approximately 56% and 33% respectively for the adaptive and classic schedules with no loss in solution quality.

The heuristic chosen for the TSSA RSMT system is based on Kruskal’s minimum-spanning tree algorithm [5]. Kruskal’s algorithm is first run to obtain the minimum-spanning tree for the n terminals using no Steiner points. Up to n - 1 Steiner points are then added in a greedy fashion from the set of possible Steiner locations to form the initial solution for the SA phase. Experimental data used for evaluating the TSSA RSMT system consists of four of the larger nets from PrimarySC2 as well as randomly generated 20 and 30 terminal networks. The results are shown in Tables 4 and 5. The average speedup over SA is approximately 40% and 30% respectively for the adaptive and classic schedules with no loss in solution quality.

The Croes heuristic [1] is used for the TSSA TSP system. Experimental data used for evaluating the TSSA TSP system consists of four well-known instances from the literature [9] ranging in size from 20 to 318 cities, as well as two randomly generated instances of size 50 and 100 cities. The results are given in Tables 6 and 7. The average speedup over SA is approximately 51% and 44% respectively for the adaptive and classic schedules with no loss in solution quality.

<table>
<thead>
<tr>
<th>Data instance (cells)</th>
<th>SA CPU time (sec)</th>
<th>SA nets cut</th>
<th>TSSA CPU time (sec)</th>
<th>TSSA nets cut</th>
<th>% CPU time decrease</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.69</td>
<td>18.8</td>
<td>0.44</td>
<td>18.8</td>
<td>36.2</td>
</tr>
<tr>
<td>500</td>
<td>7.30</td>
<td>110.0</td>
<td>3.84</td>
<td>119.3</td>
<td>47.4</td>
</tr>
<tr>
<td>833</td>
<td>18.97</td>
<td>82.1</td>
<td>6.52</td>
<td>82.6</td>
<td>65.6</td>
</tr>
<tr>
<td>1500</td>
<td>39.56</td>
<td>328.8</td>
<td>15.91</td>
<td>327.2</td>
<td>59.8</td>
</tr>
<tr>
<td>3014</td>
<td>145.41</td>
<td>222.6</td>
<td>48.67</td>
<td>228.3</td>
<td>66.5</td>
</tr>
<tr>
<td>10000</td>
<td>984.57</td>
<td>2066.9</td>
<td>386.83</td>
<td>2060.4</td>
<td>60.7</td>
</tr>
</tbody>
</table>

Table 2: Results for the TSSA VLSI-NPP system using an adaptive schedule.

<table>
<thead>
<tr>
<th>Data instance (cells)</th>
<th>SA CPU time (sec)</th>
<th>SA nets cut</th>
<th>TSSA CPU time (sec)</th>
<th>TSSA nets cut</th>
<th>% CPU time decrease</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.43</td>
<td>23.2</td>
<td>0.33</td>
<td>20.1</td>
<td>23.2</td>
</tr>
<tr>
<td>500</td>
<td>3.18</td>
<td>140.7</td>
<td>2.27</td>
<td>141.3</td>
<td>28.6</td>
</tr>
<tr>
<td>833</td>
<td>5.92</td>
<td>108.4</td>
<td>4.05</td>
<td>107.5</td>
<td>31.6</td>
</tr>
<tr>
<td>1500</td>
<td>12.91</td>
<td>398.0</td>
<td>7.91</td>
<td>400.4</td>
<td>38.5</td>
</tr>
<tr>
<td>3014</td>
<td>29.59</td>
<td>381.1</td>
<td>21.03</td>
<td>381.0</td>
<td>35.5</td>
</tr>
<tr>
<td>10000</td>
<td>140.06</td>
<td>2611.2</td>
<td>84.48</td>
<td>2616.4</td>
<td>39.7</td>
</tr>
</tbody>
</table>

Table 3: Results for a TSSA VLSI-NPP system using a classic schedule.

5. CURRENT RESEARCH EFFORTS

We are currently developing an alternative stop criterion based on the value of the offset to obtain an additional reduction in running time. The expected value of the offset is a constant throughout the majority of an SA run [9]. In the final phase, the offset quickly approaches zero. We have investigated a parameterized method of tracking significant changes in the offset from its expected value. Although we are still attempting to determine a recommended parameter value, we have tested the new stop criterion on the entire problem suite. The new stop criterion is very effective with respect to the classic cooling schedule—running times were further decreased by approximately 10% [9].

6. CONCLUSIONS

We propose a TSSA method with a formal basis for determining the temperature at which to begin the low temperature SA phase for traditional monotone cooling schedules. The results for three important optimization problems are consistently very good. On average, for adaptive SA, the
running time is cut in half; while for classic SA, the running time is reduced by one-third. Equally important is that there is on average no loss in solution quality as compared to SA.

7. REFERENCES


---

**Table 4:** Results for a TSSA RSMT system using an adaptive schedule.

<table>
<thead>
<tr>
<th>Data instance (cases)</th>
<th>SA CPU time (sec)</th>
<th>SA tree length</th>
<th>TSSA CPU time (sec)</th>
<th>TSSA tree length</th>
<th>% CPU time decrease</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>4.81</td>
<td>1554.4</td>
<td>3.82</td>
<td>1554.4</td>
<td>20.6</td>
</tr>
<tr>
<td>11</td>
<td>19.57</td>
<td>2822.0</td>
<td>12.37</td>
<td>2822.0</td>
<td>36.8</td>
</tr>
<tr>
<td>13</td>
<td>11.40</td>
<td>1950.6</td>
<td>4.33</td>
<td>1949.6</td>
<td>62.0</td>
</tr>
<tr>
<td>16</td>
<td>47.36</td>
<td>3008.4</td>
<td>19.42</td>
<td>3002.4</td>
<td>59.0</td>
</tr>
<tr>
<td>20</td>
<td>78.82</td>
<td>304.9</td>
<td>543.87</td>
<td>304.5</td>
<td>30.5</td>
</tr>
<tr>
<td>30</td>
<td>5155.76</td>
<td>359.3</td>
<td>3691.14</td>
<td>359.3</td>
<td>28.4</td>
</tr>
</tbody>
</table>

**Table 5:** Results for a TSSA RSMT system using a classic schedule.

<table>
<thead>
<tr>
<th>Data instance (cases)</th>
<th>SA CPU time (sec)</th>
<th>SA tree length</th>
<th>TSSA CPU time (sec)</th>
<th>TSSA tree length</th>
<th>% CPU time decrease</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>5.20</td>
<td>1554.2</td>
<td>4.13</td>
<td>1554.0</td>
<td>20.6</td>
</tr>
<tr>
<td>11</td>
<td>13.31</td>
<td>2822.0</td>
<td>9.20</td>
<td>2822.0</td>
<td>30.9</td>
</tr>
<tr>
<td>13</td>
<td>9.91</td>
<td>1946.8</td>
<td>6.63</td>
<td>1942.6</td>
<td>33.1</td>
</tr>
<tr>
<td>16</td>
<td>31.45</td>
<td>2999.6</td>
<td>20.43</td>
<td>2998.8</td>
<td>35.0</td>
</tr>
<tr>
<td>20</td>
<td>251.69</td>
<td>305.4</td>
<td>175.79</td>
<td>304.7</td>
<td>30.2</td>
</tr>
<tr>
<td>30</td>
<td>1276.37</td>
<td>360.2</td>
<td>898.23</td>
<td>359.9</td>
<td>29.6</td>
</tr>
</tbody>
</table>

**Table 6:** Results for the TSSA TSP system using an adaptive schedule.

<table>
<thead>
<tr>
<th>Data instance (cities)</th>
<th>SA CPU time (sec)</th>
<th>SA tour length</th>
<th>TSSA CPU time (sec)</th>
<th>TSSA tour length</th>
<th>% CPU time decrease</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0.53</td>
<td>258.6</td>
<td>0.33</td>
<td>254.3</td>
<td>37.7</td>
</tr>
<tr>
<td>42</td>
<td>5.58</td>
<td>704.6</td>
<td>2.96</td>
<td>703.7</td>
<td>47.0</td>
</tr>
<tr>
<td>50</td>
<td>8.66</td>
<td>240.3</td>
<td>4.20</td>
<td>236.4</td>
<td>51.5</td>
</tr>
<tr>
<td>57</td>
<td>12.51</td>
<td>13075.3</td>
<td>5.79</td>
<td>13133.0</td>
<td>53.7</td>
</tr>
<tr>
<td>100</td>
<td>73.69</td>
<td>316.0</td>
<td>33.25</td>
<td>307.3</td>
<td>54.9</td>
</tr>
<tr>
<td>318</td>
<td>3246.18</td>
<td>42943.7</td>
<td>1330.54</td>
<td>42835.0</td>
<td>59.0</td>
</tr>
</tbody>
</table>

**Table 7:** Results for the TSSA TSP system using a classic schedule.

<table>
<thead>
<tr>
<th>Data instance (cities)</th>
<th>SA CPU time (sec)</th>
<th>SA tour length</th>
<th>TSSA CPU time (sec)</th>
<th>TSSA tour length</th>
<th>% CPU time decrease</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0.21</td>
<td>256.6</td>
<td>0.16</td>
<td>251.1</td>
<td>23.8</td>
</tr>
<tr>
<td>42</td>
<td>1.26</td>
<td>705.4</td>
<td>0.81</td>
<td>704.8</td>
<td>35.7</td>
</tr>
<tr>
<td>50</td>
<td>1.72</td>
<td>236.3</td>
<td>1.05</td>
<td>238.0</td>
<td>39.0</td>
</tr>
<tr>
<td>57</td>
<td>2.76</td>
<td>13164.6</td>
<td>1.55</td>
<td>13106.5</td>
<td>43.8</td>
</tr>
<tr>
<td>100</td>
<td>10.09</td>
<td>326.8</td>
<td>5.01</td>
<td>320.8</td>
<td>50.3</td>
</tr>
<tr>
<td>318</td>
<td>259.24</td>
<td>43347.7</td>
<td>86.58</td>
<td>43360.8</td>
<td>66.6</td>
</tr>
</tbody>
</table>

---


