Performance of a New Annealing Schedule

Jimmy Lam+, Jean-Marc Delosme
Department of Computer Science
Yale University
New Haven, CT 06520

Abstract
A new simulated annealing schedule has been developed; its application to the standard cell placement and the traveling salesman problems results in a two to twenty-four times speedup over annealing schedules currently available in the literature. Since it uses only statistical quantities, the annealing schedule is applicable to general combinatorial optimization problems.

1. Introduction
The ground states of a complex physical system can be reached by heating the system up to some high temperature and then cooling it down slowly. The simulated annealing technique, first proposed by Kirkpatrick et al. [1], exploits this analogy to solve general combinatorial optimization problems. In this technique, the configuration space of the optimization problem is explored by a controlled hill climbing search in which the control parameter, $T$, plays the role of the temperature in a physical system. By slowly decreasing the temperature according to a properly chosen annealing schedule, one can show that the globally optimal solutions can be reached with probability one [2].

In practical applications to problems such as standard cell placement in integrated circuit layout, simulated annealing gives excellent results at the expense of massive computation time. To remedy this inefficiency, various approaches have been proposed that fall into three categories: parallel implementations of simulated annealing [3,4,5], carefully controlled move generation strategies [6] and efficient annealing schedules [2,7,8]. This paper belongs to the last category.

In 1984, White [9] proposed to group solutions according to their costs and to analyze them as groups. This approach is extended in this paper to obtain a new annealing schedule; not only solutions are grouped according to their costs, but also models are introduced to describe their properties. From these models, we arrive at the new annealing schedule and the conditions on move generation strategies that give good run-time performance.

We outline the derivation of our annealing schedule in Section 2 and discuss its application to the standard cell placement and the traveling salesman problems in Section 3. Implementation of the new annealing schedule and comparison with the annealing schedule by Huang et al. [8] are also presented in Section 3. In Section 4, we discuss practical aspects of our implementation.

2. New simulated annealing schedule
The simulated annealing heuristic is based on the observation that annealing is successful if the system is kept close to thermal equilibrium as the temperature is lowered. However, to keep the system in equilibrium at all times requires that the temperature decrements be infinitesimal; a long time would have passed before the system is frozen and annealing is stopped. From a practical standpoint, a good annealing schedule must, therefore, achieve a compromise between the quality of the final solution and the computation time. Since it is difficult to determine if a system is in thermal equilibrium, we introduce an approximate equilibrium criterion: a system is in $D$-equilibrium (deterministic equilibrium) if the $D$-condition

$$\mu(s) - \lambda \sigma(s) \leq \bar{C} \leq \mu(s) + \lambda \sigma(s)$$

is satisfied, where $\bar{C}$ is the average cost of the system, $\mu(s)$ and $\sigma(s)$ are, respectively, the steady state mean and standard deviation of the cost if the system is in thermal equilibrium at inverse temperature $s = 1/T$. (For ease of presentation, we use $s$ instead of $T$.) The user specified constant $\lambda$ controls the trade-off between the computation time and the quality of the final solution; a smaller $\lambda$ leads to a better approximation of thermal equilibrium, a higher quality of the final solution, and a longer computation time. For a given $\lambda$, we can show that our annealing schedule gives the fastest decrease in temperature while satisfying the $D$-condition.

The evolution of the average cost, $\bar{C}$, is not a function of the temperature alone; it also depends on the type of moves employed. In order to study the effect of different move generation strategies on $\bar{C}$, we need to characterize the properties of these strategies. We model the conditional probability density function of the proposed new cost, $C_+$, given the current cost, $C$, as

$$p(C_+ | C) = e^{-\beta |\Delta C|} Q(C_+, C)$$
(the move generation model), where $\Delta C = C_+ - C$ is the proposed cost change. The value of the parameter $\beta$ depends on the move generation strategy while the factor $Q(C_+, C)$ is a function of the cost density (the probability density function of the cost), $p(C)$. (For large $\beta$, $Q(C_+, C)$ is proportional to $\sqrt{p(C_+)/p(C)}$.) The factor $Q(C_+, C)$ models the belief that the frequency with which a cost is proposed depends on the number of solutions possessing that cost: the more solutions with a given cost, the more likely that cost is to be proposed. The factor $e^{-\beta \Delta C}$ models the belief that the difference between the proposed cost and the current cost influences the frequency: the closer a given cost to the current cost, the more likely it is to be proposed.

The results of a test of our model on a 100-city Traveling Salesman Problem (TSP) are displayed in Fig. 1a-1b. The five inverse temperatures at which the test was performed are indicated on the annealing curve in Fig. 1a. A typical result of the test is depicted in Fig. 1b in which the solid curve represents the computed $p(C_+ | C)$ and the histogram represents the measured $p(C_+ | C)$. The computation of $p(C_+ | C)$ is based on estimated $\beta$ and measured $Q(C_+, C)$. The interested reader is referred to [10] for the details of the experiment. From these figures, we observe that the curves agree well with the histograms throughout the entire temperature range. This suggests that our model, although simple, captures the essence of move generation strategies.

To use our model effectively, we need to compute $\beta$ and $Q(C_+, C)$. Expressions for both of these quantities can be found by introducing another model for the cost density:

$$p(C) = \sum_{i=0}^{\infty} G(s+i,v) + s_e$$

(the cost density model), where $G(s,v)$ is a gamma density function,

$$G(s,v) = \frac{v^s C^{s-1} e^{-vC}}{\Gamma(s)}.$$

Any probability density function with a bounded left tail can be represented by this model. The accuracy of the resulting representation improves as the number of terms in the model increases.

Keeping the D-condition in mind and using both models, we obtain the new annealing schedule [10]:

$$s_+ = s + \lambda \frac{4\alpha(1-\alpha)^2}{s(2-\alpha)^2 \alpha(s)}$$

where $s_+$ is the new inverse temperature and $\alpha$ is the acceptance ratio. Note that $s$ is updated after every move. The temperature decreases most rapidly when $\alpha = 0.44$. Since we want to decrease the temperature as fast as possible while satisfying the D-condition, $\alpha = 0.44$ is our target acceptance ratio.

The formula for the acceptance ratio is

$$\alpha = \frac{2\beta - s}{2s}.$$  

This expression relates the acceptance ratio to $\beta$ and, consequently, to the move generation strategy. If the move generation strategy can be modified dynamically (see Section 3 for an example), we can control move generation and, hence, $\beta$. The effect of our control can be observed from the measured acceptance ratio, $\hat{\alpha}$. If $\hat{\alpha}$ is less than 0.44, we modify the move generation strategy so that $\beta$ and, therefore, $\alpha$ increases. If $\hat{\alpha}$ is greater than 0.44, we modify the move generation strategy so that $\beta$ and, therefore, $\alpha$ decreases. Thus, the move generation strategy, the parameter $\beta$ and the measured acceptance ratio constitute a closed loop control.

The derivation of the new annealing schedule embodies a number of guidelines suggested by various authors. First, the derived target acceptance ratio of 0.44 is similar to the suggestion by Binder [12, p.11] that the magnitude of the proposed cost change should be chosen such that $\alpha \approx 0.5$. Second, the new schedule can be rewritten as

$$s_+ = s + \lambda \frac{4\alpha(1-\alpha)^2}{s(2-\alpha)^2 \alpha(s)} H(s),$$

where $H(s)$ is the specific heat. This is consistent with the belief that the larger is the specific heat, the slower should be the cooling.

3. Applications

The computational formulas for the new annealing schedule are listed in Table 1. During the initialization
Temperature updating formulas \( i = 1, 2, 3, \ldots \)

\[
\begin{align*}
\sigma_i &= \frac{1}{2\sigma(0)} \\
\sigma_{i+1} &= \frac{1}{\sigma_i^2} + \lambda - \frac{4\delta(1 - \phi)^2}{\sigma_i^4(2 - \phi)}
\end{align*}
\]

Parameter initialization formulas

\[
\begin{align*}
A &= \frac{\sigma_0^2}{C_0} \\
D &= \frac{\sigma_0^2}{C_0^2} \\
B &= \frac{1}{C} \\
E &= \frac{1}{C^2}
\end{align*}
\]

Parameter updating formulas \( i = 0, 1, 2, \ldots \)

\[
\begin{align*}
A &= \frac{f(1)a(\frac{1}{C}) - f(s)}{f(1)a(\frac{1}{C}) - f(s)} \\
D &= \frac{\sigma(1)a(\frac{1}{C}) - \sigma(s)\sigma(1)}{\sigma(1)a(\frac{1}{C}) - \sigma(s)} \\
B &= \frac{f(1)a(\frac{1}{C}) - A f(s)}{f(1)a(\frac{1}{C}) - f(s)} \\
E &= \frac{1}{C}
\end{align*}
\]

Measurement formulas \( i = t, t+1, t+2, \ldots, t \)

\[
\begin{align*}
C_t &= \frac{1}{t+1} \sum_{i=t+1}^t C_i \\
S_t &= \left( \frac{1}{t+1} \sum_{i=t+1}^t [C_i - \mu(x_i)]^2 \right)^{1/2}
\end{align*}
\]

Table 1: Computational formulas for the new annealing schedule.

The weight factors \( a \) and \( k \) in the parameter updating formulas are computed using \( a = L_a / (L_a - r) \), and \( b = L_b / (L_b - r) \), where \( L_a \) and \( L_b \) are the memory lengths of the adaptive estimators for the mean and the standard deviation of the system, respectively. Smaller memory lengths give larger weight factors which, in turn, make the system forget its history faster. Finite memory lengths are desirable because our models are imperfect. By selecting different memory lengths, we allow the model parameters to vary at different speeds as annealing proceeds.

The settings of the parameters for the TSP are \( \lambda L_a = 600, \lambda L_b = 3000, \tau = 100, k = 5 \) and the settings for the standard cell placement problem are \( \lambda L_a = 60, \lambda L_b = 300, \tau = 100, k = 5 \). The test results given below were measured on a Sun 3/280-861 with MC6881 floating point option. Unless stated otherwise, they are average results of eight executions. All programs were implemented in the C programming language.

### 3.1. Standard cell placement problem

#### 3.1.1. Test results

We tested our annealing schedule in TimberWolfSC version 4.1 [11], a standard cell placement program. Given a set of standard cells of constant height and variable width, and a net list of interconnections among cells, the objective of TimberWolfSC is to place the cells in a layout so that the total length of the interconnecting wires is minimized. The test results of eight instances of the standard cell placement problem with size (the total number of cells) ranging from 183 to 2065 are shown in Table 2. The subscripts 1 and 0 are used to indicate results from TimberWolfSC's annealing schedule and the new annealing schedule, respectively. The quantities \( T_1 \) (the speedup factor) and \( W_1 \) are defined as

\[
T_1 = \frac{t_1}{t_0} \quad \text{and} \quad W_1 = \frac{w_1}{w_0}
\]

where \( t \) represents the CPU time and \( w \) represents the total wire length. The test results for \( sda \) and \( harris \) are average results of four executions while the test results for the others are average results of eight executions. We observe that our annealing schedule gives a speedup of 1.03 to 1.66 when compared with TimberWolfSC's. The test cases \( ic \) and \( sda2 \) give a much smaller speedup than the others. This is due to two reasons. The first and the most important one is the very good initial placement of \( sda2 \). The initial temperature of TimberWolfSC is set at

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>CPU time (sec.)</th>
<th>Wire length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>( t_1 )</td>
<td>( t_0 )</td>
</tr>
<tr>
<td>example</td>
<td>183</td>
<td>719</td>
<td>437</td>
</tr>
<tr>
<td>8870</td>
<td>286</td>
<td>1463</td>
<td>881</td>
</tr>
<tr>
<td>ic</td>
<td>347</td>
<td>1839</td>
<td>1550</td>
</tr>
<tr>
<td>sda2</td>
<td>469</td>
<td>2092</td>
<td>2034</td>
</tr>
<tr>
<td>sp1</td>
<td>752</td>
<td>4491</td>
<td>3187</td>
</tr>
<tr>
<td>6565</td>
<td>800</td>
<td>4880</td>
<td>3318</td>
</tr>
<tr>
<td>sda</td>
<td>2357</td>
<td>19891</td>
<td>12412</td>
</tr>
<tr>
<td>harris</td>
<td>2965</td>
<td>29798</td>
<td>20940</td>
</tr>
</tbody>
</table>

Table 2: Comparison with TimberWolfSC's annealing schedule.
500; this is not high enough to destroy the structure of the initial placement. Consequently, a good initial solution helps TimberWolfSC give a better final placement. To test this hypothesis, we performed an experiment in which a randomized placement of $sda2$ was used as the initial solution. Instead of giving an average result of 262,318, TimberWolfSC gave an average of 207,230. Though the difference is only 2%, it is significant since our annealing schedule runs 20% to 30% faster if an average of 267,350 is considered sufficient. The second reason has to do with the aspect ratios of $ic$ and $sda2$. We observe from the $xy$ column of Table 4 that the ratios of the width of the desired placement ($x$) over the height of the desired placement ($y$) are less for $ic$ and $sda2$ than for the rest. The move generation controller, which will be discussed later, works better when the set of values it can control is larger. Since standard cells are placed on rows, their $y$-positions can only take on discrete values imposed by the $y$-positions of the rows. Their $x$-positions, however, can take on any integral value—a much larger set. Therefore, a smaller $xy$ ratio gives less freedom for the controller to work with. This hinders the operation of our annealing schedule.

We tested annealing schedules from Huang et al. [8], Aarts and Van Laarhoven [7], and Mitra et al. [2] on the TSP. The cities in the test are uniformly distributed and their number ranges from 100 to 400. The annealing schedule from Huang et al. gave significantly better results, and was used to compete with ours. Table 3 shows the test results where subscript 2 indicates results from Huang's annealing schedule. From this table, we observe that our annealing schedule gives a speedup of 0.98 to 1.61 with solutions that are 14% to 42% better than those obtained with Huang's annealing schedule.

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>CPU time (sec.)</th>
<th>Wire length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$t_2$</td>
<td>$t_0$</td>
</tr>
<tr>
<td>example</td>
<td>183</td>
<td>630</td>
<td>497</td>
</tr>
<tr>
<td>8870</td>
<td>286</td>
<td>1396</td>
<td>881</td>
</tr>
<tr>
<td>ic</td>
<td>347</td>
<td>1551</td>
<td>1550</td>
</tr>
<tr>
<td>sda2</td>
<td>469</td>
<td>2091</td>
<td>2034</td>
</tr>
<tr>
<td>spl</td>
<td>752</td>
<td>5145</td>
<td>3187</td>
</tr>
<tr>
<td>5655</td>
<td>800</td>
<td>4433</td>
<td>3318</td>
</tr>
</tbody>
</table>

Table 3: Comparison with Huang's annealing schedule.

3.1.2. Implementation details

Implementation of the new annealing schedule in TimberWolfSC involves four modifications. We replaced TimberWolfSC's annealing schedule by ours and also modified the coefficient of the penalty function, $c(i)$, the range limiter [6], and the number of bins in a row. The first modification is necessary because the cost function,

$$Cost = total\_wire\_length + c(i) \cdot penalty,$$

depends on $c(i)$, which is a function of the current iteration number, $i$. TimberWolfSC always executes 104 iterations and changes temperature after every iteration, within each iteration, a fixed number of moves depending on the problem size is proposed. Since the new annealing schedule runs faster than TimberWolfSC's annealing schedule, it executes less iterations than TimberWolfSC's. In order to arrive at similar values of $c(i)$, we need to modify the way the $c(i)$'s are computed. This was done by making an educated guess of the last iteration number for the new annealing schedule and scaling the values of $c(i)$'s accordingly. Table 4 shows the settings of the last iteration number, $i$, for both the new annealing schedule and Huang's annealing schedule.

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>$xy$</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$\lambda_2$</td>
<td>$t_2$</td>
</tr>
<tr>
<td>example</td>
<td>183</td>
<td>12.0</td>
<td>70.0</td>
</tr>
<tr>
<td>8870</td>
<td>286</td>
<td>1.4</td>
<td>10.0</td>
</tr>
<tr>
<td>ic</td>
<td>347</td>
<td>0.41</td>
<td>15.0</td>
</tr>
<tr>
<td>sda2</td>
<td>469</td>
<td>0.81</td>
<td>100.0</td>
</tr>
<tr>
<td>spl</td>
<td>752</td>
<td>1.01</td>
<td>100.0</td>
</tr>
<tr>
<td>5655</td>
<td>800</td>
<td>2.21</td>
<td>100.0</td>
</tr>
</tbody>
</table>

Table 4: Parameters for Huang's annealing schedule and the new annealing schedule.
control parameters \( r \) and \( r_V \) are adjusted after every \( r \) moves. If \( \delta \) is less than 0.44, the values of \( r \) and \( r_V \) are lowered; if \( \delta \) is greater than 0.44, the values of \( r \) and \( r_V \) are raised. The parameter \( r \) is allowed to vary from \( \text{mean cell width} + 3 \times \text{standard deviation} \) to the maximum row width, while \( r_V \) is allowed to vary from 0.75 row to the maximum number of rows.

The last modification was to change the number of bins in a row. In TimberWolfSC the total number of bins is roughly equal to the number of standard cells. Since the new annealing schedule starts at infinite temperature, all proposed moves will be accepted indiscriminately. Whenever a single cell movement is proposed, the number of empty bins can only be decreased. This is because when a cell is moved from a bin occupied by more than one cell, the number of vacant bins decreases by one. And there is no way to get a vacant bin back! Since single cell movements are proposed only when empty bins are found, the percentage of single cell movements proposed at later temperatures will be small. On the other hand, TimberWolfSC starts at an initial temperature of 500. At this temperature, not all proposed moves are accepted so that typically 10% to 20% of the bins remain empty. In order to allow the new annealing schedule to propose single cell movements at a higher rate, we increase the number of bins by decreasing the bin size. Different total numbers of bins had been tried; we settled on a ratio of 1.5 for the total number of bins to the total number of cells. This guarantees a 33% chance for single cell movements to be proposed.

Huang's annealing schedule was implemented as in [8]. The results quoted in Table 3 for this annealing schedule were obtained using TimberWolfSC's range limiter. Due to the difference in range limiters between this version of TimberWolfSC and the older version used in [8], a few maximum generation limits had been tried. We settled on a value of 5.5*\text{no of cells} \times 6, which corresponded roughly to the number of possible moves when the range limiter was used.

### 3.2. Traveling salesman problem

We also tested our annealing schedule on the TSP in which the cities are uniformly distributed on a grid of 10,000 by 10,000, and the cost is the length of the tour. In this implementation, each city maintains a list of up to 280 neighboring cities sorted according to their distances from that city in ascending order. A move is proposed by picking two cities \( A \) and \( B \), and modifying the tour as shown in Fig. 2. City \( A \) is always picked randomly while city \( B \) is picked using one of the two following methods. In the first method called \textit{uniform control}, city \( B \) is picked randomly from the first \( d \) cities on the ordered list of city \( A \). The control parameter \( d \) is allowed to decrease as a function of the inverse temperature, \( s \),

\[
d = 0.5 + \frac{N}{s} \left( \log_2(10 + \frac{1}{s \sqrt{N}}) - 1 \right)^2,
\]

where \( N \) is the total number of cities. In the second method called \textit{exponential control}, city \( B \) is the \( s \)-th entry on the ordered list of city \( A \) with

\[
d = -r \cdot \log(\text{RAND}),
\]

where \( \text{RAND} \) is a random number between 0 and 1, and \( r \), a control parameter between 2 and \( N \), is adjusted based on the measured acceptance ratio in a similar fashion as in the standard cell placement problem.

We performed tests on the TSP with the number of cities ranging from 100 to 400. The experimental results are displayed in Table 5 in which \( \delta \), representing the quality of the final solution, is the percentage of the cost above the estimated best cost. To find this estimated best cost, we carried out a sequence of careful annealings with the run-time doubled after every eight executions until the average cost was stabilized. Then, the best cost in the sequence was used as the estimated best cost. Along each column in Table 5, the speedups associated with a particular \( \delta \) as the number of cities increases are shown, while along each row the speedups associated with a particular problem size as the quality of the solution improves are shown. The speedups are defined as the ratio of the CPU time with Huang's annealing schedule using uniform control over the CPU time with the new annealing schedule using exponential control. Although our annealing schedule is 10% slower per move than Huang's, we still observe a speedup of up to 24 for the 400-city TSP!

<table>
<thead>
<tr>
<th>Size</th>
<th>Speedup</th>
<th>Speedup</th>
<th>Speedup</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>2.09</td>
<td>3.39</td>
<td>6.00</td>
<td>8.35</td>
</tr>
<tr>
<td>200</td>
<td>2.54</td>
<td>4.00</td>
<td>7.50</td>
<td>10.61</td>
</tr>
<tr>
<td>300</td>
<td>2.90</td>
<td>4.38</td>
<td>10.86</td>
<td>24.20</td>
</tr>
<tr>
<td>400</td>
<td>3.37</td>
<td>7.77</td>
<td>17.61</td>
<td>&gt;21.00</td>
</tr>
</tbody>
</table>

Table 5: Comparison with Huang's annealing schedule for the TSP.

To isolate the effect of exponential control, we experimented on both annealing schedules with uniform and exponential controls. The speedups, computed as the CPU time using uniform control over the CPU time using exponential control on the new annealing schedule, are displayed in Table 6, while the speedups, computed as the CPU time of Huang's annealing schedule over the CPU time of the new annealing schedule when both were using exponential control, are displayed in Table 7.
Table 6: Comparison of uniform and exponential controls (with the new annealing schedule) for the TSP.

<table>
<thead>
<tr>
<th>Size</th>
<th>Speedup δ = 3.6%</th>
<th>Speedup δ = 2.9%</th>
<th>Speedup δ = 2.2%</th>
<th>Speedup δ = 1.5%</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1.94</td>
<td>2.02</td>
<td>2.06</td>
<td>2.26</td>
</tr>
<tr>
<td>200</td>
<td>2.19</td>
<td>2.26</td>
<td>2.30</td>
<td>2.40</td>
</tr>
<tr>
<td>300</td>
<td>2.55</td>
<td>2.63</td>
<td>2.70</td>
<td>2.80</td>
</tr>
<tr>
<td>400</td>
<td>2.55</td>
<td>2.63</td>
<td>2.70</td>
<td>2.80</td>
</tr>
</tbody>
</table>

Table 7: Comparison with Huang’s annealing schedule enhanced with exponential control for the TSP.

<table>
<thead>
<tr>
<th>Size</th>
<th>Speedup δ = 3.6%</th>
<th>Speedup δ = 2.9%</th>
<th>Speedup δ = 2.2%</th>
<th>Speedup δ = 1.5%</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1.95</td>
<td>1.99</td>
<td>2.04</td>
<td>2.21</td>
</tr>
<tr>
<td>200</td>
<td>2.19</td>
<td>2.02</td>
<td>2.06</td>
<td>2.26</td>
</tr>
<tr>
<td>300</td>
<td>2.55</td>
<td>2.26</td>
<td>2.36</td>
<td>2.70</td>
</tr>
<tr>
<td>400</td>
<td>2.55</td>
<td>2.26</td>
<td>2.36</td>
<td>3.01</td>
</tr>
</tbody>
</table>

A few maximum generation limits for Huang’s annealing schedule had also been tried; we settled on a value of $N \times d/2$ which corresponded to the number of possible moves when uniform control was used.

4. Practical considerations

We observe from Table 1 that our annealing schedule uses floating point computations heavily. To reduce the number of floating point operations, we approximate our annealing schedule by computing the new inverse temperature after every ten steps instead of one:

$$s_{t+10} = s_t + 10 \alpha \frac{4 \alpha (1 - \alpha)^2}{s_t^2 (2 - \alpha) s_t^2 (s_t)}.$$  

This speeds up our annealing schedule by a few percents without degrading the quality of the final solutions.

We also pre-compute the value of exponentials and logarithms. Their values are looked up when needed. Since the evaluation of exponentials and logarithms occurs quite often, a significant portion of run-time is typically saved.

Care must be taken in designing exponential controls; the number of possible moves at any time should not be too small. Consider the TSP example. If the value of $\alpha$ in the exponential control is too close to 0, the value of $s_t$ will be restricted to a small set. This is undesirable since the number of possible moves is so small that annealing may stop prematurely.

Acknowledgements

The authors would like to acknowledge Carl Sechen and Kai Win Lee from the Department of Electrical Engineering at Yale University for many helpful discussions on TimberWolfSC. This research was supported by the Army Research Office under contract DAAL03-86-K-0158 and by the Office of Naval Research under contract N00014-85-K-0461.

References


