Performance of a New Annealing Schedule

Jimmy Lam†, Jean-Marc Delosme††
†Department of Computer Science
††Department of Electrical Engineering
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*

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

is satisfied, where $\overline{C}$ 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 $s \equiv 1/T$. (For ease of presentation, we use s instead of T.) The user specified constant λ controls the trade-off between the computation time and the quality of the final solution; a smaller λ leads to a better approximation of thermal equilibrium, a higher quality of the final solution, and a longer computation time. For a given λ, we can show that our annealing schedule gives the fastest decrease in temperature while satisfying the D-condition.

The evolution of the average cost, $\overline{C}$, 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 $\overline{C}$, 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

$$p(C_+ \mid C) = e^{-\beta \mid \Delta C \mid} Q(C_+, C)$$

(the move generation model), where $\Delta C = C_+ - C$ is the proposed cost change. The value of the parameter β 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 β, $Q(C_+, C)$ 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 is 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 $p(C_+ \mid C)$ and the histogram represents the measured $p(C_+ \mid C)$. The computation of $p(C_+ \mid C)$ is based on estimated β 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.

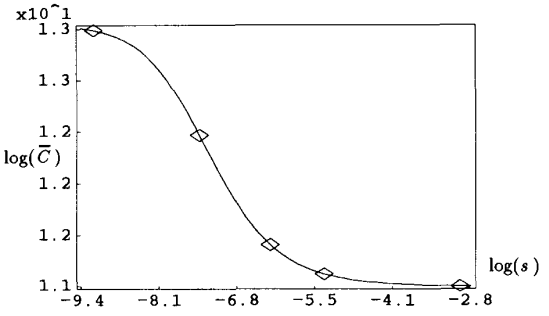

Figure 1a: Inverse temperatures at which the test was performed.

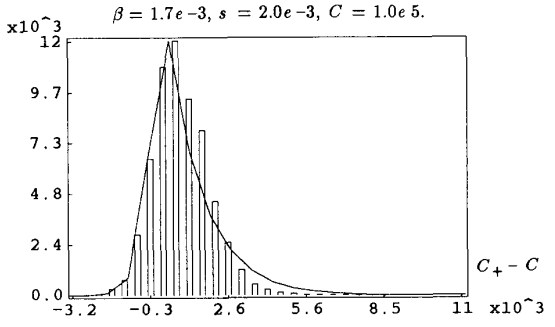

Figure 1b: Testing the move generation model.
Measured $p(C_+ \mid C)$, histogram.
Computed $p(C_+ \mid C)$, solid curve.

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

$$p(C) = \sum_{i=0}^{n} h_i G(u+i, v) + g_o$$

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

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

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]:

$$s_+ = s + \lambda \frac{4\alpha(1-\alpha)^2}{s^2(2-\alpha)^2\sigma^3(s)},$$

where s_+ is the new inverse temperature and α 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

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

This expression relates the acceptance ratio to β 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, β. 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 β and, therefore, α increases. If $\hat{\alpha}$ is greater than 0.44, we modify the move generation strategy so that β and, therefore, α decreases. Thus, the move generation strategy, the parameter β 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

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

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 $t = 1, 2, 3, \ldots$

$$s_1 = \frac{1}{2\sigma(0)} \qquad s_{t+1} = s_t + \lambda \frac{4\hat{\alpha}(1-\hat{\alpha})^2}{s_t^2(2-\hat{\alpha})^2\sigma^3(s_t)}$$

$$\mu(s_t) = \frac{1}{As_t + B} \qquad \sigma(s_t) = \frac{1}{Ds_t + E}$$

Parameter initialization formulas

$$A = \frac{\bar{S}_0^2}{\bar{C}_0^2} \qquad D = \frac{\bar{S}_0}{\bar{C}_0}$$

$$B = \frac{1}{\bar{C}_0} \qquad E = \frac{1}{\bar{S}_0}$$

Parameter updating formulas $i = 0, \tau, 2\tau, \ldots$

$$A = \frac{f(1)f(\frac{s}{\bar{C}}) - f(s)}{f(1)f(s^2) - f(s)f(s)} \qquad D = \frac{g(1)g(\frac{s}{\bar{S}}) - g(s)g(\frac{1}{\bar{S}})}{g(1)g(s^2) - g(s)g(s)}$$

$$B = \frac{f(1)f(\frac{1}{\bar{C}}) - Af(s)}{f(1)} \qquad E = \frac{g(1)g(\frac{1}{\bar{S}}) - Dg(s)}{g(1)}$$

$$f(x) = \sum_{i=0}^{t} x_i a^i \qquad g(x) = \sum_{i=0}^{t} x_i b^i$$

Measurement formulas $i = t-\tau+1, t-\tau+2, \ldots, t$

$$\bar{C}_t = \frac{1}{\tau}\sum_{i=t-\tau+1}^{t} C_i \qquad \bar{S}_t = \left[\frac{1}{\tau}\sum_{i=t-\tau+1}^{t} [C_i - \mu(s_i)]^2\right]^{1/2}$$

Table 1: Computational formulas for the new annealing schedule.

phase, the inverse temperature, s_0, is set to 0. The system is allowed to run until it is randomized and accurate estimations of the mean, $\bar{C}_0$, and the standard deviation, $\bar{S}_0$, can be made. These estimates are used to compute the initial values of the parameters A, B, D, and E using the initialization formulas. Then, the new inverse temperature is computed using the temperature updating formulas after every move, and the parameters A, B, D, and E are adjusted using the parameter updating formulas after every τ moves. This process is repeated until $\bar{C}$ remains unchanged for the last $k\tau$ moves. At this point, the system is considered frozen and annealing is stopped. Note that the measured acceptance ratio, $\hat{\alpha}$, is recomputed after every τ moves.

The weight factors a and b in the parameter updating formulas are computed using $a = L_a / (L_a - \tau)$, and $b = L_b / (L_b - \tau)$, 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-s8 with MC68881 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 2965 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

Name	Size	CPU time (sec.)			Wire length		
		t_1	t_0	T_1	w_1	w_0	W_1
example	183	719	437	1.65	218049	216285	1.01
8870	286	1463	881	1.66	181621	177399	1.02
ic	347	1839	1550	1.19	83232	83027	1.00
sda2	469	2092	2034	1.03	262318	262427	1.00
sp1	752	4491	3187	1.41	986564	974308	1.01
5655	800	4880	3318	1.47	958758	946914	1.01
sda	2357	19891	12412	1.60	2165850	2179570	0.99
harris	2965	29798	20940	1.42	2006360	1832440	1.09

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 267,250. Though the difference is only 2%, it is significant since our annealing schedule runs 20% to 30% faster if an average of 267,250 is considered sufficient. The second reason has to do with the aspect ratios of *ic* and *sda2*. We observe from the **x:y** 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 **x:y** 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.

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.

The range limiter was also modified. In TimberWolfSC, rows are divided into rectangular regions called bins. A move is proposed by first picking a cell, say cell A, randomly. Then a window, whose size is independent of the cell and is equal to $2*mean\_cell\_width + 6*standard\_deviation$ by 3 rows, is centered around cell A. A bin is picked randomly within that window. If the bin contains no cell, a single cell movement is proposed; if the bin contains one or more cells, a cell is picked randomly within that bin and a pairwise exchange is proposed. Instead of picking a bin this way, we modified the range limiter by allowing the window size to change dynamically and picking a bin according to the formulas,

$$d_x = \pm\, r_x * \log(\text{RAND}) \quad \text{and} \quad d_y = \pm\, r_y * \log(\text{RAND}).$$

Here, RAND is a random number between 0 and 1, d_x and d_y are, respectively, the x and y distances of the new bin from cell A, and r_x and r_y are the control parameters. If bins are picked this way, the probability that a bin is picked is proportional to e^{-d_x/r_x} in the x-direction, and e^{-d_y/r_y} in the y-direction; therefore, bins that are closer to cell A have a higher probability of being picked. The

Name	Size	CPU time (sec.)			Wire length		
		t_2	t_0	T_2	w_2	w_0	W_2
example	183	630	437	1.44	251567	216285	1.16
8870	286	1396	881	1.58	202731	177399	1.14
ic	347	1521	1550	0.98	98842	83027	1.19
sda2	469	2091	2034	1.03	371500	262427	1.42
sp1	752	5145	3187	1.61	1235140	974308	1.27
5655	800	4433	3318	1.34	1154030	946914	1.22

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,

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

depends on $c(i)$, which is a function of the current iteration number, i. TimberWolfSC always executes 104

Name	Size	x:y	Parameters			
			λ_2	i_2	λ_0	i_0
example	183	12.0:1	70.0	104	0.03	54
8870	286	1.4:1	10.0	104	0.014	54
ic	347	0.4:1	15.0	104	0.01	60
sda2	469	0.8:1	100.0	104	0.013	90
sp1	752	1.0:1	100.0	104	0.009	54
5655	800	2.2:1	100.0	104	0.008	54

Table 4: Parameters for Huang's annealing schedule and the new annealing schedule.

control parameters r_x and r_y are adjusted after every τ moves. If $\hat{\alpha}$ is less than 0.44, the values of r_x and r_y are lowered; if $\hat{\alpha}$ is greater than 0.44, the values of r_x and r_y are raised. The parameter r_x is allowed to vary from *mean_cell_width* + 3**standard_deviation* to the maximum row width, while r_y 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**no_of_cells*, 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 250 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 *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,

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

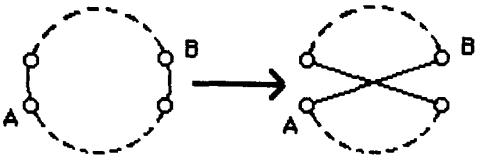

Figure 2: How to modify a tour in the TSP.

where N is the total number of cities. In the second method called *exponential control*, city B is the d^{th} entry on the ordered list of city A with

$$d = -r*\log(\text{RAND}),$$

where RAND is a random number between 0 and 1, and r, a control parameter between 2 to 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 δ, 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 δ 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!

Size	Speedup			
	$\delta = 3.6\%$	$\delta = 2.9\%$	$\delta = 2.2\%$	$\delta = 1.5\%$
100	2.09	3.39	6.00	8.35
200	2.54	4.09	7.50	10.61
300	2.90	4.38	10.86	24.20
400	3.37	7.77	17.61	>21.00

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. We

Size	Speedup			
	$\delta = 3.6\%$	$\delta = 2.9\%$	$\delta = 2.2\%$	$\delta = 1.5\%$
100	1.74	3.27	3.43	3.87
200	1.37	1.75	2.23	2.76
300	1.94	1.76	3.00	5.00
400	1.37	1.87	2.82	4.90

Table 6: Comparison of uniform and exponential controls (with the new annealing schedule) for the TSP.

Size	Speedup			
	$\delta = 3.6\%$	$\delta = 2.9\%$	$\delta = 2.2\%$	$\delta = 1.5\%$
100	1.95	1.99	2.04	2.21
200	2.19	2.02	2.06	2.26
300	2.55	2.26	2.36	2.70
400	2.55	2.27	2.80	3.01

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

observe that the use of exponential control speeds up the new annealing schedule by a factor of up to five. However, even when the same control method is employed with both annealing schedules, our annealing schedule still out-performs Huang's by a factor of up to three. For comparisons with other heuristics like Lin and Kernighan [13] for the TSP and Fiduccia and Mattheyses [14] for the graph partition problem, the interested reader is referred to [15].

A few maximum generation limits for Huang's annealing schedule had also been tried; we settled on a value of $N*d/2$ 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:

$$s_{t+10} = s_t + 10\lambda \frac{4\alpha(1-\alpha)^2}{{s_t}^2(2-\alpha)^2\sigma^3(s_t)}.$$

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 r in the exponential control is too close to 0, the value of d 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

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

[2] D. Mitra, F. Romeo and A.L. Sangiovanni-Vincentelli, "Convergence and Finite-time Behavior of Simulated Annealing," *Proceedings of the 24th Conference on Decision and Control*, 761-767, 1985.

[3] S. Kravitz and R. Rutenbar, "Multiprocessor-Based Placement by Simulated Annealing," *Proceedings of the 23rd IEEE Design Automation Conference*, 567-573, 1986.

[4] E. Aarts, F. de Bont, E. Habers and P. van Laarhoven, "Parallel Implementations of the Statistical Cooling Algorithm," *INTEGRATION, the VLSI journal*, Vol. 4, 209-238, 1986.

[5] P. Banerjee and M. Jones, "A Parallel Simulated Annealing Algorithm for Standard Cell Placement on a Hypercube Computer," *Proceedings of the IEEE International Conference on Computer-Aided Design*, 34-37, 1986

[6] C. Sechen and A. Sangiovanni-Vincentelli, "The TimberWolf Placement and Routing Package," *IEEE Journal of Solid-State Circuits*, Vol. 20, No. 2, 510-522, 1985.

[7] E. Aarts and P. Van Laarhoven, "Statistical Cooling: a General Approach to Combinatorial Optimization Problems," *Philips Journal of Research*, Vol. 40, No. 4, 193-226, 1985.

[8] M. Huang and A. Sangiovanni-Vincentelli, "An Efficient General Cooling Schedule for Simulated Annealing," *Proceedings of the IEEE International Conference on Computer-Aided Design*, 381-384, 1986.

[9] S. White, "Concepts of Scale in Simulated Annealing," *Proceedings of the IEEE International Conference on Computer Design*, 646-651, 1984.

[10] J. Lam and J-M. Delosme, "An Adaptive Annealing Schedule," Report 8608, Department of Electrical Engineering, Yale University, Sept. 1986.

[11] C. Sechen and K.W. Lee, "TimberWolfSC Version 4.1 Program Source," 1987.

[12] K. Binder, *Monte Carlo Methods in Statistical Physics*, 2nd Edition, Springer-Verlag, 1986.

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

[14] C. Fiduccia and R. Mattheyses, "A Linear-Time Heuristic for Improving Network Partitions," *Proceeding of the 19th IEEE Design Automation Conference*, 175-181, 1982.

[15] J. Lam, "An Efficient Simulated Annealing Schedule," Ph.D. Dissertation, Department of Computer Science, Yale University, Fall 1988.