EXPERIMENTS WITH SIMULATED ANNEALING*

Surendra Nahar — University of Minnesota
Sartaj Sahni — University of Minnesota
Eugene Shragowitz — Control Data Corporation

ABSTRACT
The performance of simulated annealing is compared to that of other Monte Carlo methods for optimization. Our experiments show that these other methods often perform better than simulated annealing.

Key words and phrases
Simulated annealing, Monte Carlo method, heuristic, optimization

1. INTRODUCTION

In 1953 Metroplois et al., [METR53], proposed a Monte Carlo method to simulate the equilibrium state of a collection of atoms at any given temperature T. This method is based on Boltzmann's distribution of atomic configurations at any temperature. By Boltzmann's law, the probabilty of any configuration i of atoms is given by $e^{-E(i)/(k_b T)}$ where $E(i)$ is the energy associated with configuration i and k_b is Boltzmann's constant. Hence, the most probable configurations at any given temperature are those with the lowest energy. Since the number of possible configurations is extremely large, a heuristic to find a configuration with low energy was devised by Metropolis et al.

The Metropolis method is very readily adapted to the solution of arbitrary combinatorial optimization problems. Suppose we wish to minimize the goal function $h(i)$ subject to the constraints C. Here, i is a vector that denotes a possible solution to the optimization problem. The Metropolis method translates to the following scheme:

```
Step 1   Let $i$ be a random feasible solution (i.e., $C(i)$ is
         satisfied) to the given problem. Set *counter* = 0
Step 2   Let $j$ be a feasible solution that is obtained from
         $i$ as a result of a random perturbation. This per-
         turbation may, for example, be a pairwise ex-
         change or may involve a random change in a sin-
         gle element of $i$, etc.
Step 3   If $h(j)-h(i) < 0$, then [$i = j$, update best solution
         found so far in case $i$ is best, *counter* = 0, go to
         Step 2].
Step 4   [$h(j)-h(i) \geq 0$] If *counter* $\geq n$ then stop.
         Otherwise, $r$ = *random*
         if $r < e^{-(h(j)-h(i))/(k_b T)}$ then [$i = j$, *counter* = 0]
         else [*counter* = *counter* + 1]
         go to Step 2.
```

The Metropolis adaptation combined with Kirkpatrick's several temperature method is called *simulated annealing*. To use simulated annealing on any optimization problem, we need a temperature schedule T_1, ..., T_k. In practice (see [KIRK83] and [GOLD84]), the Boltzmann's constant is combined with the T_is and a schedule for $k_b T_i$ is obtained. Let Y_i denote $k_b T_i$. Henceforth, we shall use the term *temperature* to refer to a Y_i rather than to just T_i. In [KIRK83], the Y_is were chosen to form an exponentially decreasing sequence. Specifically, for the circuit partition problem, the schedule used was $Y_1 = 10$, $Y_i = 0.9*Y_{i-1}$, $2 \leq i \leq 6$. In [GOLD84] the Y_i were chosen to be 25 unformly distributed points in some interval $(0, \tau)$.

*The research reported here was supported in part by the Office of Naval Research under contract N00014-80-C-0650, by the National Science Foundation under grant DCR-8305567, and by Control Data Corporation under grant 84M301.

2. PREVIOUS EXPERIMENTAL WORK

Kirkpatrick et al. ([JEPS83], [KIRK83], and [VECC83]) have reported significant success in using simulated annealing for a variety of optimization problems that arise in the design automation of electronic circuits. While these experiments used simulated annealing essentially as described here, they often varied in the terminating criterion (i.e., the criterion used to determine when equilibrium at the given temperature has been achieved) for the Metropolis adaptation. For instance, in the experiments of [KIRK83], the Metropolis adaptation was terminated when either a sufficient number of random perturbations had been accepted or when the total number of perturbations generated had become sufficiently large.

There is a deficiency in the experiments of [KIRK83]. No attempt is made to compare annealing with other proven heuristic methods for the problems considered. Golden and Skicism, [GOLD84], conducted rather extensive experiments comparing simulated annealing with other heuristic methods known for the traveling salesperson problem. Their results indicate that simulated annealing does not perform as well as some of the sophisticated heuristics developed for this problem. In fact, using an evenly distributed 25 temperature schedule, simulated annealing required about 20 to 60 times the computing time required by the heuristic of Stewart, [STEW77]. Furthermore, the solutions produced by simulated annealing were significantly inferior to those obtained using the heuristic of [STEW77]. [GOLD84] also compares simulated annealing with the 2-opt heuristic of [LIN73]. The 2-opt heuristic of [LIN73] is given enough starting random tours to make its run time comparable to that of simulated annealing. Of 10 instances that were tried, simulated annealing did better that [LIN73] just once. The remaining nine times, [LIN73] produced better solutions.

[GREE84] develops a method to improve the run time performance of annealing at low temperatures. The method proposed trades computer time with computer space. In fact, the authors themselves state that the memory cost is great. [ROME84a and b], [LUND83], and [GEM83] show that under certain constraints, simulated annealing obtains optimal solutions asymptotically. Some guidelines on choosing the highest and lowest temperatures in an annealing schedule are provided in [WHIT84].

3. SOME VARIATIONS

Annealing has a sound theoretical basis in the physical domain. No such basis for its application to arbitrary combinatorial problems exists. Some attempts to provide such a theoretical justification appear in [ROME84 a and b], [GREE84], [LUND83], and [GEM83]. All of these however make limiting assumptions.

Simulated annealing is just one of many possible Monte Carlo methods that accept perturbations that result in an increase in objective function value. With the exception of [COHO83a and b], there appears to be no work reporting on the relative performance of these other Monte Carlo methods. Cohoon and Sahni ([COHO83a] and [COHO83b]) made two modifications to the Metropolis-Kirkpatrick method. The heuristic used by them is described below:

```
Step 1   Let $i$ be a random feasible solution to the given problem.
Step 2   Continue to perturb $i$ until no perturbation results in a decrease in $h$. At this time, $i$ is said to be locally optimal with respect to the perturbation strategy. For instance if pairwise interchange is being used, then pairwise interchanges are performed on $i$ until no pairwise interchange reduces $h(i)$.
Step 3   Update the best solution found so far, if $i$ is best.
Step 4   $r = random$
         if $r < g(h(i))$ then make a random perturbation to $i$ and go to Step 2. Otherwise, stop.
```

The notable differences are:

1. Perturbations that increase the objective function value are considered for acceptance only after a local optima has been reached.
2. The possibility of using an arbitrary function in place of the negative exponential function derived from annealing is introduced.

Combining the efforts of [KIRK83] and [COHO83a and b], we can formulate the two strategies given in Figures 1 and 2. k denotes the number of Y_is in the annealing schedule. The method of [KIRK83] corresponds to Figure 1 with $g_{temp}(h(i),h(j)) = e^{-(h(j)-h(i))/Y_{temp}}$. The method of [COHO83a and b] corresponds to Figure 2 with $g_{temp}(h(i),h(j)) = c*h(i)$ and $k = 1$.

```
Step 1   Let $i$ be a random feasible solution to the given problem. $temp = 1$. $counter = 0$
Step 2   Let $j$ be a feasible solution that is obtained from $i$ as a result of a random perturbation.
Step 3   If $h(j)-h(i) < 0$, then [$i = j$, update best solution found so far in case $i$ is best, $counter = 0$, go to Step 2].
Step 4   [$h(j)-h(i) \geq 0$] If $counter \geq n$ then [if $temp = k$ then stop else [$temp = temp + 1$, $counter = 0$, go to Step 2]]
         Otherwise, $r = random$
         if $r < g_{temp}(h(i),h(j))$ then [$i = j$, $counter = 0$]
         else [$counter = counter + 1$]
         go to Step 2.
```

Figure 1

In this paper, we experiment with the two strategies of Figures 1 and 2. Specifically, we attempt to determine if one of the two strategies is consistently superior to the other and whether the choice of the g_{temp} functions materially affects performance. The specific g functions that we consider are:

1. **Metropolis**
   $k = 1$, $g_1(h(i),h(j)) = e^{-((h(j)-h(i))/Y_1}$
2. **Six Temperature Annealing**
   $k = 6$, $g_{temp}(h(i),h(j)) = e^{-((h(j)-h(i))/Y_{temp}}$, $1 \leq temp \leq 6$

```
Step 1   Let $i$ be a random feasible solution to the given problem. $temp = 1$. $counter = 0$
Step 2   Continue to perturb $i$ until no perturbation results in a decrease in $h$.
Step 3   Update the best solution found so far, if $i$ is best.
Step 4   if $counter \geq n$ then [if $temp = k$ then stop else [$temp = temp + 1$, $counter = 0$]]
Step 5   $counter = counter + 1$, $r = random$
         let $j$ be the result of a random perturbation to $i$
         if $r < g_{temp}(h(i),h(j))$ then [$i = j$, go to Step 2]
         go to Step 4
```

Figure 2

3. **g=1**
   $k = 1$, $g_1(h(i),h(j)) = 1$
   When the strategy of Figure 1 is used, a straightforward implementation of this results in a random walk through the solution space. To prevent this, it is implemented such that a perturbation that increases the energy is accepted only if a sufficiently long sequence of perturbations has failed to yield a configuration of lower energy. Specifically, this g was implemented by us in the following way when the strategy of Figure 1 is used. Each time a random perturbation reduces the energy, a counter is set to zero. Each time the energy is increased the counter is incremented by 1. However, the higher energy configuration does not become the starting point for further perturbations until the counter becomes 18. At this time, the counter is reset to 1.
   When the strategy of Figure 2 is used, no special considerations are needed to implement this g.
4. **Two Level g**
   $k = 2$, $g_1 = 1$, $g_2 = .5$
5. **Linear**
   $k=1$, $g_1(h(i),h(j)) = Y_1*h(i)$
6. **Quadratic**
   $k=1$, $g_1(h(i),h(j)) = Y_1*h(i)^2$
7. **Cubic**
   $k=1$, $g_1(h(i),h(j)) = Y_1*h(i)^3$
8. **Exponential**
   $k=1$, $g_1(h(i),h(j)) = (e^{h(i)/Y_1}-1)/(e-1)$
9. **Six Temperature Linear**
   $k=6$, $g_{temp}(h(i),h(j)) = Y_{temp}*h(i)$, $1 \leq temp \leq 6$
10. **Six Temperature Quadratic**
    $k=6$, $g_{temp}(h(i),h(j)) = Y_{temp}*h(i)^2$, $1 \leq temp \leq 6$
11. **Six Temperature Cubic**
    $k=6$, $g_{temp}(h(i),h(j)) = Y_{temp}*h(i)^3$, $1 \leq temp \leq 6$
12. **Six Temperature Exponential**
    $k=6$, $g_{temp}(h(i),h(j)) = (e^{h(i)/Y_{temp}}-1)/(e-1)$, $1 \leq temp \leq 6$
13. **Linear Difference**
    $k=1$, $g_1(h(i),h(j)) = Y_1/(h(j)-h(i))$
14. **Quadratic Difference**
    $k=1$, $g_1(h(i),h(j)) = Y_1/(h(j)-h(i))^2$
15. **Cubic Difference**
    $k=1$, $g_1(h(i),h(j)) = Y_1/(h(j)-h(i))^3$
16. **Exponential Difference**
    $k=1$, $g_1(h(i),h(j)) = (e^{Y_1/(h(j)-h(i))}-1)/(e-1)$
17. **Six Temperature Linear Difference**
    $k=6$, $g_{temp}(h(i),h(j)) = Y_{temp}/(h(j)-h(i))$, $1 \leq temp \leq 6$
18. **Six Temperature Quadratic Difference**
    $k=6$, $g_{temp}(h(i),h(j)) = Y_{temp}/(h(j)-h(i))^2$, $1 \leq temp \leq 6$
19. **Six Temperature Cubic Difference**
    $k=6$, $g_{temp}(h(i),h(j)) = Y_{temp}/(h(j)-h(i))^3$, $1 \leq temp \leq 6$
20. **Six Temperature Exponential Difference**
    $k=6$, $g_{temp}(h(i),h(j)) = (e^{Y_{temp}/(h(j)-h(i))}-1)/(e-1)$, $1 \leq temp \leq 6$

To obtain a fair comparison of the above methods, we need to control the amount of computer time each is allowed.

Unless this is done, we will be unable to say whether better performance results from the additional time given one method or from an inherent superiority of the method. In all of our experiments we restricted each method to complete its task in the same amount of time.

Method 3 is the easiest to use. It involves none of the constants Y_i and all the available computer time is to be spent on a single g. To use method 4, we need to determine how much time is to be spent using g_1 relative to time spent using g_2. In order to use any of the other methods, we need to determine the values of the constants Y_i. Additionally, when $k>1$, we need to determine the amount of time to be spent at each temperature. We might expect that the performance of a method depends quite heavily on the choice of the Y_is and also on the amount of time spent at each Y_i. The validity of this expectation is easily verified by experiment.

4. OPTIMAL LINEAR ARRANGEMENT

4.1. Problem Definition

We consider two forms of the optimal linear arrangement problem. One form is the *net optimal linear arrangement* (NOLA) problem. In this form, we are given n circuit elements (cells, boards, chips, etc) and connectivity information. We are required to obtain a linear ordering of these n elements that minimizes the number of nets that cross between any pair of elements that are adjacent in this ordering. The maximum number of nets that cross between any pair of adjacent elements is called the *density* of the linear arrangement. In the NOLA problem, we are required to find a linear arrangement with minimum density. This problem is identical to the board permutation problem studied in [GOTO77], and [COHO83a]. The problem also arises in the placement of standard cells and gate arrays [KANG83], and in the ordering of via columns in single row routing [RAGH84] and [TING78]. The second form that we consider is actually a special case of NOLA. In this special case, each net connects exactly two circuit elements. The connectivity information and circuit elements now form a graph. The resulting problem is called the *graph optimal linear arrangement problem* (GOLA).

4.2. Graph Optimal Linear Arrangement (GOLA)

4.2.1. Determining The Temperatures

The best Y_is to use depend on the following:

1. The amount of time spent at each temperature.
2. Whether the strategy of Figure 1 or Figure 2 is being used.
3. The actual instance to be solved.
4. The actual form of the g_i functions (i.e., which of the 20 classes of g listed in Section 3 is in use).

Since it is impractical to determine the best Y_is for each combination of instance characteristics, strategy type (Figure 1 or Figure 2), g function class, and amount of time spent at each temperature, we attempt to find the best Y_is for each g using a randomly generated set of instances and the strategy of Figure 1. In our experiments, this was done by first generating 30 random GOLA instances. Each instance consisted of 15 circuit elements and 150 two pin nets. These 30 instances were solved using each of the 18 g classes that use Y_is. If a g class uses k Y_is, then the time spent at each temperature was limited to $\lceil 5/k \rceil$ seconds. The solution for each instance was obtained using pairwise interchange beginning with a random linear arrangement of the circuit elements. Each g class used the same initial arrangement. All our experiments were conducted on a VAX 11/780. All programs were written in Pascal. The Y_is that gave the best results on the above test data were used for further experimenting.

4.2.2. Comparative Results

Table 4.1 shows the total reduction in g values for each of the 20 g function classes when the strategy of Figure 1 is used and each class is permitted 6, 9, and 12 seconds per instance. These results were obtained using the same 30 instances as used earlier to obtain the best Y_is. As can be seen, in most cases, performance improved as more time was made available. The few exceptions can be explained by the randomness in the algorithms. The sequence of perturbations attempted for a six second run is different from that attempted for a 9 second run. This in turn is different from that attempted for a 12 second run. This difference explains the apparent anomalies exhibited in many of our forthcoming tables in which performance may decrease as time is increased.

Best performance is exhibited by six temperature annealing, $g = 1$, and cubic difference. While cubic difference performed better than the six temperature annealing on two of the three times, the difference in performance cannot be regarded as significant. Similarly, even though six temperature annealing performed better for all three times than $g = 1$, the performance difference is not significant. The extremely good performance of $g = 1$ is exciting as this g requires no decisions regarding temperatures.

g function	6 sec	9 sec	12 sec
Goto	601	-	-
[COHO83a]	474	505	519
Metropolis	533	558	569
Six Temperature Annealing	601	632	652
$g = 1$	598	605	646
Two level g	546	524	582
Linear	464	495	520
Quadratic	447	493	500
Cubic	451	462	477
Exponential	488	461	535
6 Linear	488	494	524
6 Quadratic	455	486	502
6 Cubic	457	511	502
6 Exponential	475	510	513
Linear Diff	587	591	614
Quadratic Diff	515	527	541
Cubic Diff	618	626	654
Exponential Diff	597	599	617
6 Linear Diff	524	579	615
6 Quadratic Diff	528	506	546
6 Cubic Diff	586	591	620
6 Exponential Diff	552	574	631

Table 4.1 30 instances, 15 elements, 150 nets

Table 4.1 also gives the reductions obtained using some of the heuristics proposed earlier for this problem. The heuristic of Goto, [GOTO77], constructs the linear arrangement left to right. It begins with the most lightly connected element and places this at the leftmost position of the linear arrangement. Let S be the set of nets in the elements already placed. Let i be an element not yet placed, and let T be the nets in the remaining elements not yet placed. The next element, i, to be placed is chosen such that ST is minimum over all choices for i. This next element is added to the right of the arrangement so far obtained. This heuristic required about 6 sec of CPU time per instance. As can be seen, this performs as well as the best g functions when these are limited to the about the same computing time. When additional computing time is provided to the Monte Carlo methods, their performance becomes better than that of the Goto heuristic.

The heuristic proposed in [COHO83a] is a Monte Carlo heuristic that uses the g function:

$$g_1(density) = \min(density/(m+5), 0.9)$$

where m is the number of nets in the instance (150 for our test instances). Cohoon and Sahni used this g function together with the startegy of Figure 2. They experimented with several different interchange heuristics such as pairwise and single exchange. They concluded that from their set of heuristics, the best was one that started with the result of [GOTO77] and used a single exchange method coupled with the above g function. To get the results for our table, we simply used the above g function together with the strategy of Figure 1 and pairwise interchange. Presumably, the reductions in density would have been greater had we used the best heuristic reported in [COHO83a]. This best heuristic would however have required more than 6 sec per instance to complete its task. Hence, our results here should be regarded only as illustrating the performance of their g function when used together with pairwise interchange.

4.2.3. Coupling Monte Carlo and GOTO

Because of the extremly good performance of the heuristic of [GOTO77] relative to that of the Monte Carlo method, we attempted to determine the effect of using the linear arrangement produced by this heuristic as the starting arrangement for the Monte Carlo heuristics. The total improvements obtained for the initial 30 instance test set are summarized in Table 4.2(a).

1. The performance of each g class (except for $g = 1$ and two level g) is quite sensitive to the temperature schedule used. The best temperature schedule varies from one class to another. In fact, even for the same g class, the best temperature schedule depends on the specific instance being solved.
2. When the amount of CPU time available is small, simple greedy heuristics can be expected to perform as well as any of the Monte Carlo methods. The strategy of Figure 2 cannot be used at this time because of the excessive time to find a local optima.
3. When slightly more time is available, starting from a good initial solution yields slightly better results than starting from a random initial solution.
4. When enough time is made available and the right choice between the strategies of Figures 1 and 2 is made, all g classes may be expected to perform the same.
5. The best performance is exhibited by six temperature annealing, $g = 1$, cubic difference, and six temperature cubic difference. When starting with Goto's arrangement, the performance of six temperature exponential difference was comparable (better in some cases) to that of the four methods just mentioned.

g function	6 sec	9 sec	12 sec	Figure 1	Figure 2	6 sec	9 sec	12 sec	6 sec	9 sec	12 sec
[COHO83a]	7	38	41	651	727	368	-	-	6	6	6
Metropolis	35	35	40	682	692	183	245	281	4	4	4
Six Temperature Annealing	78	61	79	739	701	251	251	311	8	0	12
$g = 1$	45	49	52	736	735	303	319	288	11	11	11
Two level g	5	9	9	642	703	388	388	388	3	3	2
Linear Diff	38	46	59	709	738	288	313	312	2	2	2
Quadratic Diff	20	18	30	656	736	318	321	323	0	0	0
Cubic Diff	31	43	76	741	729	207	237	283	2	2	2
Exponential Diff	41	43	62	726	735	212	289	338	11	20	20
6 Linear Diff	41	56	55	719	738	306	309	311	2	0	2
6 Quadratic Diff	26	35	39	647	734	316	319	314	2	2	2
6 Cubic Diff	79	87	91	743	731	210	237	282	2	2	2
6 Exponential Diff	55	78	86	727	739	215	295	336	10	4	2
		(a)			(b)	307	315 (c)	323		(d)	

30 instances, 15 elements, 150 nets

Table 4.2

The sum of the densities of the starting arrangements is 2594–601 = 1993. The best improvement is obtained by six temperature cubic. However, this improvement is less than 5%. Looking at the 12 sec improvements, we see that the improvements obtained by Six temperature annealing, cubic difference, six temperature cubic difference, and six temperature exponential difference are comparable.

4.2.4. Figure 1 vs Figure 2

The last set of the experiments we conducted with GOLA involved the strategy of Figure 2. This time, perturbations that increase energy are accepted only after a local optima has been reached. The results are shown in Table 4.2 (b). Each method was given 3 minutes per instance. The set of 30 instances used earlier was used for this test. For comparison purposes, the results of a 3 minute per instance run using the strategy of Figure 1 are also included.

The shift from the strategy of Figure 1 to that of Figure 2 has mixed results. Significant improvements occur for [COHO83a], two level g, and quadratic difference. In all the performance of 9 of the 13 g classes improved as a result of this change in strategy. The time per instance was chosen to be 3 minutes as it takes about 20 seconds to find a local optima. It was felt that 3 minutes will provide an adequate exercise of the Monte Carlo technique. Surprisingly, when the better of the two strategies is considered for each g class, the performance difference between any pair of g classes is at most 6%. The performance of all 13 classes is about the same.

4.2.5. Summary

On the basis of our experiments with GOLA, we draw the following conclusions:

6. The easiest g class to use is $g = 1$. The performance of this is almost as good as that of any other class. For the other g classes (except for two level g), the choice of temperatures can have a significant impact on performance. The CPU time needed to determine good temperatures is more profitably spent by running $g = 1$ for a longer time.

4.3. Net Optimal Linear Arrangement (NOLA)

4.3.1. Experimental Results

Our experiments with the NOLA problem ignored the g function classes 5 through 12 because of their poor performance on the GOLA instances. The temperatures used for this problem are the same as those used for the GOLA problem. The results obtained for a test set of 30 NOLA instances are shown in Table 4.2 (c). Each instance in this test set has 15 circuit elements and 150 nets. The sum of the densities of the starting random arrangements is 4254. The improvements obtained are a little less than 10%. $g = 1$ is the only g class that has a performance better than the heuristic of [GOTO77]. Its performance is almost 30% better than that of six temperature annealing.

When the linear arrangement produced by [GOTO77] is used as the starting arrangement, *none* of the 13 Monte Carlo methods is able to obtain a significant improvement. This may be becuse of the near optimality of the Goto arrangements. Table 4.2 (d) shows the small improvement obtained by the various methods.

4.3.2. Summary

The relative performance of the various methods is somewhat different for NOLA than with GOLA. The conclusions to be drawn are:

1. Only $g = 1$ has a better performnce than the heuristic of [GOTO77]. This remains true even if the Monte Carlo methods are given twice as much time as required by [GOTO77].
2. The performance of six temperature annealing is significantly inferior to that of $g = 1$. The performance of the different g functions (excluding $g = 1$) is comparable.
3. When we begin with the arrangement obtained by [GOTO77], exponential difference is the stellar performer. It outperforms its nearest rivals (six temperature annealing and $g = 1$) by a factor of 2. However, no method makes a significant improvement over the initial arrangement.

The striking commonality between these conclusions and those made for GOLA is in the good performance of $g = 1$.

5. CONCLUSIONS

For the problems we have considered, $g = 1$ is recommended for the following reasons:

1. It involves no user decisions. There are no parameters to be determined before this g can be used. Of the gs considered. it is bv far the easiest to use.
2. Its performance is uniformly near the best performance observed by any of the gs considered.

Experiments were also performed using the Circuit Partition and Traveling Salesperson problem . Results may be found in [NAHA84].

6. REFERENCES

[COHO83a] J. Cohoon and S. Sahni, Heuristics for the board permutation problem, *Proceedings 1983 IC CAD Conference*, Sept 1983.

[COHO83b] J. Cohoon and S. Sahni, Heuristics for the circuit realization problem, *Proceedings 20th Design Automation Conference*, June 1983.

[GEMA83] D. Geman and S. Geman, Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images.

[GOLD84] B. Golden and C. Skiscim, Using simulated annealing to solve routing and location problems, University of Maryland, College of Business Administration, Technical Report, Jan. 1984.

[GOTO77] S. Goto, I. Cederbaum, and B.S. Ting, "Suboptimal Solution of the Backboard Ordering with Channel Capacity Constraint", *IEEE Trans. Circuits and Systems*, Nov. 1977, pp. 645-652.

[GREE84] J. Greene and K. Supowit, Simulated annealing without rejected moves, Proceedings ICCD, Oct. 1984, pp 658-663.

[JEPS83] D. Jepsen and C. Gelatt Jr., Macro placement by Monte Carlo annealing, *Proceedings 1983 IC CAD Conference*, Sept 1983, pp. 495-498.

[KANG83] S. Kang, Linear ordering and application to placement, *Proceedings 20th Design Automation Conference*, 1983, pp 457-464.

[KIRK83] S. Kirkpatrick, C. Gelatt, Jr., and M. Vecchi, Optimization by simulated annealing, *Science*, Vol 220, No 4598, May 1983, pp. 671-680.

[LIN73] S. Lin and B. Kernighan, An effective heuristic for the traveling salesman problem, *Operations Research*, Vol 21, pp. 498-516, 1973.

[LUND83] M. Lundy and A. Mees, Convergence of the annealing algorithm, University of Cambridge, 1983.

[METR53] N. Metropolis, A. Rosenbluth, A. Teller, and E. Teller, Equation of state calculations by fast computing machines, *Jr. Chem. Phys.*, Vol 21, p. 1087, 1953.

[NAHA84] S. Nahar ,S. Sahni and E. Shragowitz ,Experiments with simulated annealing, University of Minnesota,Minneapolis,Technical report # 84-36,1984.

[RAGH84] R. Raghavan and S. Sahni, The complexity of single row routing, *IEEE Trans. On Circuits and Systems*, Vol CAS-31, No 5, May 1984, pp. 462-471.

[ROME84a] F. Romeo, A. Vincentelli, and C. Sechen, Research on simulated annealing at Berkeley, Proceedings ICCD, Oct. 1984, pp 652-657.

[ROME84b] F. Romeo and A. Vincentelli, Probabilistic hill climbing algorithms: Properties and applications, University of California, Berkeley, UCB/ERL M84/34, 1984.

[STEW77] W. Stewart, A computationally efficient heuristic for the travelling salesman problem, *Proceedings of the 13th Annual Meeting of Southeastern TIMS*, Myrtle Beach, S.C., pp 75-83, 1977.

[TING78] B. Ting and E. Kuh, An approach to the routing of multilayer printed circuit boards, *Proc. IEEE Symp. On Circuits and Systems*, pp. 902-911, 1978.

[WHIT84] S. White, Concepts of scale in simulated annealing, Proceedings ICCD, Oct 1984, pp 646-651.

[VECC83] M. Vecchi and S. Kirkpatrick, Global wiring by simulated annealing, *IEEE Trans. On Computer Aided Design*, Vol CAD-2, No 4, Oct. 1983, pp 215-222.