View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Purdue E-Pubs

Purdue University
Purdue e-Pubs

Department of Electrical and Computer Engineering Technical Reports

Department of Electrical and Computer Engineering

12-1-1989

Backtracking IC Placement Algorithm

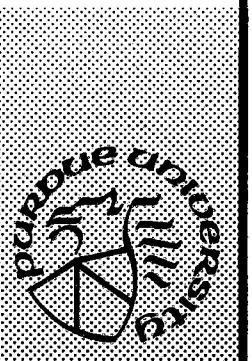
Yi Cheng
Purdue University

Robert H. Fujii
Purdue University

Follow this and additional works at: https://docs.lib.purdue.edu/ecetr

Cheng, Yi and Fujii, Robert H., "Backtracking IC Placement Algorithm" (1989). *Department of Electrical and Computer Engineering Technical Reports.* Paper 691.
https://docs.lib.purdue.edu/ecetr/691

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.



Backtracking IC Placement Algorithm

Yi Cheng
Robert H. Fujii

TR-EE 89-70
December, 1989

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

Backtracking IC Placement Algorithm

ABSTRACT

A new algorithm for integrated circuit (IC) layout placement is introduced. As in simulated annealing, it allows uphill movements but in a more restrictive manner; thus, the search for an optima is more directed. Experiments on standard cell placement have shown that the average convergence time is faster than the simulated annealing algorithm while achieving similar results.

1. Introduction

The goal of IC layout placement is to reduce the wiring length of cell interconnections so that the layout area is minimized. A fairly complete bibliography of previous placement techniques can be found in [1, 2]. As indicated in [1], most previous approaches failed to compete with a human designer because the placement problem is NP-complete [3] and traditional greedy algorithms often end at local minima.

Recently, the simulated annealing algorithm [4] became very attractive for the placement problem because it can converge to a global optima with probability one. Successful implementations of this algorithm have been reported in [1]. However, simulated annealing theoretically needs an infinite number of iterations to reach the global optima. A carefully planned cooling schedule is necessary when only a finite number of iterations are available [5, 6, 7]. Adjusting the cooling schedule experimentally may be time consuming and tedious in applications.

In this report, we propose a new approach for IC layout placement. This method searches for the best solution among all the randomly generated configurations in the placement process. The algorithm has the following features:

1) The algorithm does not require a process schedule such as the cooling schedule used in simulated annealing. It starts with a randomly chosen initial placement and searches for the best solution until a specified stop criterion is satisfied. The user specified stop criterion can be used to control the quality of the solution within the allotted computer execution time.

2) In order to avoid being trapped in a local optima, the algorithm maximizes the distance between the current placement and a worse placement result generated during an earlier iteration; this permits uphill movements in the placement process.

3) The average computation time complexity is lower than the simulated annealing with comparable results.

In this report, we will model the behavior of the algorithm using a finite Markov chain transition matrix and show that the algorithm converges to a global optima quickly. Some basic definitions are given in section 2 and the basic algorithm is discussed in section 3. Theoretical discussions appear in section 4. In section 5, we show some computer simulation results and examine the effectiveness of our approach. Section 6 gives conclusions.

2. Definitions

To describe our algorithm, we need the following definitions:

Backtracking Chain

It is a sequence of placement configurations with decreasing wiring cost. For example, assume we have 6 different placement configurations in a stack S_p = {c6, c5, c4, c3, c2, c1}, with c6 at top of the stack and with corresponding wiring costs C = {100, 110, 120, 170, 150, 90}. In this case, S1 = {c6, c5, c4, c3} is a backtracking chain; however neither S2 ={c2 c3 c4} nor S3 = {c3, c2, c1} are backtracking chains. In S1, the **tail of chain,** c3, corresponds to the placement configuration with the largest wiring cost, while the **head of chain,** c6, has the smallest wiring cost. The length of the backtracking chain is the difference in cost between its head and tail.

Peak Distance

It is the wiring cost difference between the head and tail of the longest backtracking chain $S_{longest}$.

Current Placement Configuration

It is the most recently accepted layout placement configuration. From this configuration, we search for a better layout placement configuration.

Valley Distance

It is the wiring cost difference between the current placement configuration and a past placement configuration with the lowest cost. This cost difference is expressed as:

$$V_0 = Valley\ distance = C_i - C_0$$

Where C_i = wiring cost of current placement configuration, and C_0 = the lowest wiring cost of past accepted placement configurations.

Cell

It is a standard cell which will be placed by the algorithm.

Backtracking Chain Length K

It represents the depth of the backtracking stack; thus, the maximum number of elements in the stack is K. Experiments have shown a value of K between 3 and 5 is optimal.

3. The Algorithm

In a real placement problem, we need to consider wire length, overlap penalty, critical path weighting and several other parameters that will affect the final layout. However, to simplify the discussion, we will only use wire length cost to gauge the quality of a placement result. Using the definitions given in section 2, we can now describe our algorithm as follows:

1) A backtracking stack S_p is used to keep track of placement configurations which have been rejected; the size of S_p is equivalent to the number of entries in the stack. Initially, the stack is empty, the peak distance is set to zero, and the value of V_0 is set to infinity. The algorithm begins with a randomly chosen placement configuration.
2) A candidate placement configuration is generated randomly through either a) interchange of cell positions, b) single cell displacement, or c) cell orientation changes.
3) The wiring cost of the candidate placement configuration is pushed onto the stack S_p.
4) If the candidate placement configuration has lower wiring cost than the current placement configuration, it is accepted as the new current configuration, stack S_p is cleared, the value of V_0 is updated and we go back to step 2. However, If the candidate configuration has higher wiring cost than the current placement

configuration, the following steps are carried out to see if it should be accepted.

5) If the size of S_p is less than K or if it is equal to K but the S_p does not form a backtracking chain, the candidate configuration is rejected and we go back to step 2.

6) If size of the stack S_p is equal to K and S_p forms a backtracking chain, the following procedures are carried out:

- update V_{peak}
- fd = wiring cost of candidate configuration - C_i
- if (α*$fd < V_0$ and $fd < S_{longest}$) {
- accept the candidate configuration;
- clear S_p;
- go back to step 2).
- } else {
- reject the candidate configuration;
- go to step 2);
- }

α is a user specified variable which controls the acceptance ratio. Its value ranges between 0 and 1. When α is equal to 1, the algorithm turns into a greedy algorithm. Usually, α is selected to be small at the beginning of the search process; then we increase it by a small fraction every M iterations (M is proportional to the number of cells in the layout, e.g. M=25*cell_number). It functions like the temperature parameter in the simulated annealing algorithm and can be used to control the acceptance ratio of newly generated configurations. In our experiments, α is set to 0.002 initially and increased 2% every 25*Nc iterations (Nc is the number of cells in the layout).

7) In above iterations, we will check a stop criterion as follows: if in M iterations (M is the same as specified above) less than β% of newly generated configurations are accepted, the algorithm terminates. β is set to 2% in our experiments.

The basic idea behind this algorithm is to simultaneously maximize the distance between the current configuration and a worse placement configuration (approximated by $S_{longest}$) and minimize the distance between the current configuration and the best configuration (approximated by V_0).

4. Theoretical Discussion

The Markov chain has been used as an effective model to analyze the behavior of the simulated annealing algorithm [7, 8, 10]. We will also use the Markov chain to analyze our algorithm and compare it to simulated annealing.

Let S={1, 2, 3, ... , N} be a finite set and let f be a function defined on S; the elements in S denote the state number in the solution space. The simulated annealing approach, is modeled as an inhomogeneous Markov process with state space S and with entries of the transition matrix defined as:

$$P_{ij} = \begin{cases} q(i,j)\exp[-(f(j)-f(i))^{+}/T(t)] & \text{for } j \neq i \\ 1 - \sum_{k \neq i} P_{ik} & \text{for } j = i \end{cases} \quad (1)$$

Where $T(t)$ is a positive function converging to zero as t goes to infinity. $f(i)$ is the wiring cost of placement configuration i. $q(i,j)$ is the probability of generating configuration j from configuration i. It has been proven that the annealing process will converge to the lowest cost configuration with probability one [5, 7]. To simplify the analysis and comparison, we assume that the elements of set S have cost corresponding to their index 1, 2, ..., N. We construct the transition matrix with lower cost elements occupying the lower index position. Equation (1) can be rewritten as

$$P_{ij} = \begin{cases} \frac{1}{N}\exp[-(j-i)^{+}/T(t)] & \text{for } j \neq i \\ 1 - \sum_{k \neq i} P_{ik} & \text{for } j = i \end{cases} \quad (2)$$

Since we assume that there is an equal chance to generate any placement configuration, q(i,j) equals $\frac{1}{N}$. The Markov process corresponding to equation 2 can easily be simulated with a computer program. Using the mean first passage matrix M [9], one can estimate the convergence time by looking at the matrix entry $M(i,j)$ which denotes the mean number of iterations required for transition from state i to state j. Using the limiting vector A discussed in [9], one can estimate the probability of reaching a global optima. Table 1 shows the results from mean first passage matrix and limiting vector on a problem with state space S = 40 and q(i,j)=$\frac{1}{40}$. In order to simulate a realistic case (the case when there is a local minima), we set q(1,2)=q(2,1)=0 which specifies that configuration 2 cannot go to the global optima (configuration 1) directly, An up-hill movement is necessary before the placement configuration can change from the local minima (configuration 2) to the global

optima (configuration 1).

Table 1: Simulation Results Modeling Behavior of Simulated Annealing Algorithm.

Optimal Probability	# of iteration
68%	177
92%	935
99.2%	11830

From Table 1, it can be seen that the computation time increases dramatically when the optimal probability (the probability for a configuration to be a global optima at the end of iterations) is close to 1. When N is very large in real implementations, the required computation time could be extremely long.

Using set S and the same assumptions that we used to construct equation 2, the transition matrix for the Markov process in our algorithm can be written as (see Appendix for derivation):

$$P_{ij} = \begin{cases} d(i,j)\dfrac{C_K^{N-j}}{N^K} & \text{for } j>i \\ \dfrac{1}{N} & \text{for } j<i \\ 1-\sum_{m\neq i} P_{im} & \text{for } j=i \end{cases} \tag{3}$$

Where

$$\text{if } (j-i < j-1 \ .and.\ j-i \le N-j) \tag{4}$$

$$d(i,j) = 1$$

$$else \quad d(i,j) = 0$$

Simulations on a 40x40 matrix are again carried out to observe the behavior of our algorithm. Table 2 shows the results of simulations.

Table 2: Simulation Results Modeling Behavior of Best Searching Algorithm.

Optimal Probability	# of iteration	K
100%	43	2
100%	48	3
100%	77	4

It can be noted from Table 2 that the results are far superior to those shown in Table 1. The computational complexity is of order O(N).

It should be noted that in order to obtain the results shown in Table 2, we have made a very unrealistic assumption that we know from the outset the wiring cost of the best placement. This assumption becomes reasonable in the later stages of the search process as we get closer and closer to the global optima. Since a large number of iterations are still performed in the later stages of the search process, our algorithm can still be advantageous (compared to simulated annealing) by reducing the number of iterations. This is shown in the next section by two real placement examples.

5. Experimental Results

We used two real placement examples to compare our algorithm against simulated annealing. The results are shown in Tables 3 and 4.

Table 3: Actual IC Placement Results, Number of Cells=30.

	Backtracking	Simulated Annealing
Wire Length Reduction	31%	25%
CPU Time	25.1 sec.	68.6 sec.

Table 4: Actual IC Placement Results, Number of Cells=90.

	Backtracking	Simulated Annealing
Wire Length Reduction	61%	62%
CPU Time	68.0 sec.	250.3 sec.

The results shown in Tables 3 and 4 represent the average of several simulation runs. We can see that the new algorithm achieves the same results as simulated annealing in substantially shorter time.

6. Conclusions

A new algorithm for IC placement has been introduced. It allows uphill movements to avoid being trapped in a local minima. Its advantages are : 1) no fine tuning of algorithm necessary; 2) on the average, it converges to a global optima at faster rate than simulated annealing; and 3) uses simple arithmetic computations, (i.e., no exponential computations). The algorithm can be useful in other combinatorial optimization problems where the computation complexity is NP complete.

Appendix

To derive equation 3, we need two assumptions.

1) The probability to generate any configuration from a given configuration is the same, i.e., it is uniformly distributed.
2) We assume that the backtracking chain has length K.

A new configuration which decreases the wiring cost, is always accepted, i.e., the probability of selecting a configuration with lower wiring cost is simply equal to the probability of generating it, $\frac{1}{N}$.

A new configuration which increases the wiring cost, is not accepted unless a) a backtracking chain is found, and b) the increased cost is less than the valley distance and the peak distance. By considering a) and b) together we get equation 3.

To find a backtracking chain, we need to pick K configurations with decreasing wiring cost. From assumption 1), the upper bound of the probability to generate K configurations with decreasing wiring cost is:

$$\frac{C_K^{N-j}}{N^K} = \begin{cases} \left[\frac{(N-j)!}{K!(N-j-K)!}\right]\left[\frac{1}{N^K}\right] & \text{for } N-j \geq K \\ 0 & \text{for } N-j < K \end{cases} \tag{5}$$

The condition that the increasing cost must be less than the valley distance can be idealized as

$$j - i < j - 1 \tag{6}$$

Which is satisfied by all i except i = 1. Note that j-1 here actually is the distance from current configuration to the best configuration.

The condition that the increasing cost must be less than the peak distance can be idealized as

$$j - i \leq N - j \tag{7}$$

Combining equations 5 through 7, we get equation 3.

References

[1] Sechen, Carl. "VLSI Placement and Global Routing Using Simulated Annealing." *Kluwer Academic Publishers.* (1988).

[2] Hildebrandt, T. "An Annotated Placement Bibliography." *ACM SIGDA Newsletter* (Dec. 1985): 12-21.

[3] Garey, M. and D. Johnson. "Computers and Intractability: A Guide to the Theory of NP-Completeness. *San Francisco: Freeman* (1979).

[4] Kirkpatrick, S., C. Gelatt, and M. Vecchi. "Optimization by Simulated Annealing." *Science* 220/4598 (1983): 671-680.

[5] Hajek, B. "Cooling Schedules for Optimal Annealing." *Mathematics of Operations Research* 13, (1988): 311-329.

[6] Laarhoven, P.J.M. Van and E.H.L. Aarts. "Simulated Annealing: Theory and Applications," *Reidel, Dordrecht.* (1987).

[7] Aarts, E.H.L. and J. Korst. "Simulated Annealing and Boltzmann Machines A stochastic Approach to Combinatorial Optimization and Neural Computing." *John Wiley and Sons* (1989).

[8] Chiang, Tzuu-shuh and Chow YunShyong. "On the Convergence Rate of Annealing Processes." *SIAM J. Control and Optimaization.* (Nov.,1988): Vol. 26, no. 6, 1455-1470.

[9] Kemeny, John G and Snell, J. Laurie. "Finite Markov Chains." *D. Van Nostrand Company, Inc* (1960).

[10] D. Mitra, F. Romeo, and A. Sangiovanni-Vincentelli. "Convergence and Finite-Time Behavior of Simulated Annealing" *Proc. 24th Conf. on Decision and Control* (1985):761-767.