Quality-Time Tradeoffs in Simulated Annealing for VLSI Placement

Srilata Raman and Benjamin Wah
Center for Reliable and High Performance Computing
Coordinated Science Laboratory
University of Illinois
1101 West Springfield Avenue
Urbana, IL 61801
srilata@aquinas.csl.uiuc.edu

Abstract

The paper presents a model to characterize the relationship between the best solution (incumbent) found by an iterative algorithm (simulated annealing) and the time spent in achieving it. The target application has been chosen to be the placement of cells on a VLSI chip. The model is used to achieve a tradeoff between solution quality and time spent. This gives an idea of the time that the iterative algorithm should be terminated when the marginal gain in solution quality is smaller than the marginal increase in cost (or time) spent. Nonlinear regression analysis is used to predict the decrease in time with respect to improvement in solution quality. Experimental results on benchmark circuits are presented to show the errors of run-time prediction as compared to a static prediction. The method adopted is very general and can be extended to a variety of applications that use iterative algorithms.

Index Terms.

Regression analysis, simulated annealing, solution quality, VLSI placement, VLSI routing.

1. Introduction

Automatic layout in VLSI design refers to the transformation of the logical design of a circuit to a physical design. It has two components: placement and routing. Placement is responsible for positioning the components of the circuit on the layout surface, while routing involves connecting the components such that a prespecified set of design rules is complied with. The goals of placement and routing are intertwined: a good placement simplifies the job of a router, while a poor placement might result in a situation in which not all the connections can be made in the given area. Placement can be done to satisfy a variety of objective functions specified by the designer. In this paper, the objective of placement is to minimize the estimated length of the interconnection wiring. Although placement and routing are interdependent, the computational complexity of the tasks makes it imperative to deal with the two separately (both are known [2] to be NP-hard). In this paper, the emphasis is on placement although the concepts can be extended to routing as well.

Placement has been addressed in two different ways: constructive and iterative. Constructive placement methods start with a partial placement (none or some components have assigned positions) and build on it to produce the complete placement using some heuristic. Iterative methods start with a complete placement (very often generated by a constructive method) and perform heuristic-based perturbations to produce better placements.

Substantial research has gone into developing various algorithms for solving the placement problem in the past, resulting in a plethora of techniques. The emphasis in all such attempts has been to improve the quality of solution obtained, very often in terms of the interconnection lengths. No attempt has been made to study how the quality of solutions varies with respect to time. The tradeoff is particularly useful for iterative methods, since such a knowledge can be utilized to terminate the algorithm when it is noted that the solution quality will not improve substantially with time. It is with this intent in mind that the work reported in this paper has been carried out.

The placement technique dealt with in this paper is based on simulated annealing. This is an iterative method and has been demonstrated to give good results [4]. The experimental results have been obtained using the TimberWolfe package that does placement and routing using simulated annealing. The experiments are for the gate array layout technology, but can be extended to other styles of layout as well. The circuits have been taken from the benchmarks suggested for placement and routing algorithms.

The main contribution of this paper is the proposal and empirical verification of an exponential model that helps to characterize the variation of solution quality with time. Using this model, an analysis of the tradeoff in solution quality and time has been carried out that lends useful insight into the performance of the annealing algorithm. A study of this nature has not been reported in the literature. The method presented in this

Research supported by Joint Services Electronics Program Contract N00014-90-J-1270.

Proceedings Computer Software and Applications Conference, IEEE, 1991

paper can be applied to other iterative schemes as well. It affords a means to utilize the time resource in an intelligent and productive manner.

The paper is organized in the following fashion. The next section provides a brief overview of the simulated annealing algorithm. Section 3 details the motivation for choosing simulated annealing as the iterative algorithm and placement as the target application. Section 4 deals with tradeoff studies. In Section 4.1, the proposed model is formulated, and experimental results to corroborate it are presented in the subsequent section. Experimental results on quality-time tradeoff are presented in Section 4.3. Conclusions and future course of this research are highlighted in Section 5.

2. Simulated Annealing

Simulated annealing is a technique for solving combinatorial optimization problems [1]. The main feature of this technique is that it permits *hill-climbing* moves, *i.e.*, those moves that increase the cost function being optimized. Thus, if the configuration space of the target problem has lots of solutions, this feature helps in attaining the global optima.

```
procedure SA {
        T = $T_0$ ;
        S = $S_0$ ;
        while (stopping-criterion is not satisfied) {
          while (inner-loop-criterion is not satisfied) {
            s = generate( S ) ;
            if (accept(c(s),c(S),T) is true)  S = s ;
          }
          T = update(T) ;
        }
  }

accept (c(s),c(i),T) {
        $\Delta$c = c(s) − c(i) ;
        r = random(0,1) ;
        if (r < f($\Delta$c,T))
            return(1) ;
        else
            return(0) ;
  }
```

Figure 1. Simulated annealing algorithm

The generic structure of the simulated annealing algorithm [3] is shown in Figure 1. The terminology used in the above algorithm is as follows. The term $c(i)$ refers to cost of state i; T_0 is the initial temperature; S_0 is the initial state; S is the current state; and T is current temperature. There are three essential components to the simulated annealing algorithm: the function **generate** generates new configuration states, the function **accept** decides whether the new state is acceptable, and the function **update** switches to a new temperature. The **inner-loop-criterion** decides how many perturbations are to be attempted at each temperature, while the **stopping-criterion** decides when to stop the annealing process. The cost function is chosen to be the estimated interconnection wire length. The acceptance of a state is governed by the function f defined by

$$f(\Delta c, T) = \min(1, e^{(-\Delta c/T)}) . \qquad (2.1)$$

The temperature is updated using the relation,

$$T_{new} = \alpha T_{old}, \qquad 0 < \alpha < 1 . \qquad (2.2)$$

The inner-loop criterion, the stopping criterion and the temperature updates together constitute the annealing schedule. We will not go into the details of all these components in this paper, as a comprehensive study is available in the reference [4].

3. Motivations

The mathematical model developed in this paper holds for the simulated annealing technique used to solve the placement problem for VLSI circuits. The reason for choosing this technique is that it is a method that gives complete solution to the problem if terminated before it reaches completion. In this way, no extra work need be done even if the algorithm is stopped midway. Also, simulated annealing is based on the premise that if more time is spent in searching the configuration space by way of an extended annealing schedule, the chances of reaching an optimal solution are higher. This makes it an ideal candidate for studying the variation of the solution quality with time. Such a study has largely been ignored since the emphasis has all along been to optimize the objective function for placement, oftentimes at the expense of time spent, barring of course the studies based on solving the problem on parallel architectures.

Placement has been chosen as the target application mainly because it has been the most popular application for testing the usefulness of the simulated annealing technique. Consequently, a number of implementations have been carried out, and good documentation on the implementations is available. In this study, the well known TimberWolfe package [4] has been used for the experiments. The other reason for choosing placement as the target problem is that it is a highly compute-intensive task, specially so for the current crop of millions of transistors on a single chip. It thus becomes imperative to study the tradeoff of placement quality and time spent. For instance, if it is concluded that the solution quality improves only marginally with time after a certain time period, then it is perhaps better not to spend any more time attempting to arrive at better solutions. This approach would help reduce the turnaround time for chip design. This, in essence, has been the chief motivation for this study.

It is worthwhile at this point to note that all iterative algorithms work within an implicit time frame that is guided by the number of iterations and moves carried out during the course of the algorithm. Thus there is an implicit tradeoff between the number of iterations (or time) the designer is willing to spend and the final quality of result. In this paper, we have discussed a systematic approach to study such a tradeoff in Simulated Annealing.

The term *incumbent* will be used frequently in the course of the ensuing discussion, so a definition at this point would be in order. As stated earlier, the cost function in the TimberWolfe implementation is the estimated wire lengths of all the interconnections. The solution quality is reflected by the value of this cost function, the smaller the value the better the placement; that is, the higher the quality of the placement. The value of the cost function during the annealing process fluctuates in an attempt to arrive at the smallest value. Simulated Annealing being a hill-climbing mechanism tends to shoot past the minima found during the early part of the search. In this paper, we keep track of the best solution found so far in the *incumbent*; this is done without affecting the performance of the annealing process. The incumbent is updated whenever a smaller value of cost function is found by the annealing algorithm. This also ensures a solution better than or equal to that of the traditional annealing process.

The model to be presented next is based on the behavior observed for the incumbent. Our analysis of this behavior is aimed at answering two questions. The first is whether the behavior of the incumbent can be used to predict its value after a specified time. The second is whether to terminate the process when only marginal improvement in the incumbent value with time is detected. The tradeoff is thus measured with respect to the value of the incumbent.

4. Quality-Time Tradeoffs

The salient aspect of any iterative algorithm is an improvement in solution quality with time. If it is found that the solution quality does not improve substantially with time, perhaps the algorithm can be terminated much before the regular completion. The descriptions in this section deal with the tradeoff of solution quality and time spent. In order to carry out this objective, a formal relationship between the solution quality and the time spent in achieving it needs to be established. The solution quality, Q, is defined as the reciprocal of the incumbent value. The approach adopted to arrive at the proposed model describing this relationship is elucidated in the following section. Once the relationship is known, different heuristics can be chosen depending on the tradeoff desired. The conditions for such a heuristic to be applicable are derived in Section 4.3, along with experimental results demonstrating the quality-time tradeoff for one such heuristic.

4.1. Formulation of the Model

The chief aim of our proposed model is to characterize the improvement in solution quality and construct an abstraction that can be used to predict the behavior of the solution quality with time. The abstraction could be done by either analysis or measurement. Simulated annealing, being a stochastic process, a mathematical analysis of the behavior of its solution is often restricted by the assumptions made. The other option is to go for an empirical study that provides a more realistic assessment of the behavior. A possible disadvantage of the latter approach is that the results are limited by the experiments performed. We choose to carry out an empirical study in this paper.

Statistical analysis of the TimberWolfe algorithm for carrying out placement of cells for test circuits reveals that the relationship between the incumbent values and the time spent is nonlinear. The incumbent values are found to be scattered around an exponential curve. This observed feature motivated us to conjecture that the incumbent obeys an exponential model for all circuits. In order to prove this conjecture, tests were conducted on benchmark circuits. As will be obvious later, the results reveal that this conjecture is true based on the experiments performed. The exponential model that fits the incumbent curves most closely is empirically found to be of the following form.

$$inc_i = c + e^{(-a\, t_i + b)} + \varepsilon_i \tag{4.1}$$

Here, the value of the incumbent at time t_i is inc_i. The parameters of the regression model are a, b, and c, and ε_i is the additive error term. This model is nonlinear in its parameters and time. Eq. (4.1) can also be expressed as

$$inc_i = f(t_i, \gamma) + \varepsilon_i\,, \; where \;\; \gamma = [a \;\; b \;\; c]^T\,. \tag{4.2}$$

In order to determine *good* estimators of the regression parameters, a, b, and c, the method of *least squares estimation* is employed to minimize the sum of squared errors, SSE, for n sample observations.

$$\text{SSE} = \sum_{i=1}^{n} (inc_i - f(t_i, \gamma))^2\,. \tag{4.3}$$

The estimates of γ which minimize SSE for the given sample observations (t_i, inc_i), are the least squares estimates and are denoted by

$$\gamma^* = [a^* \;\; b^* \;\; c^*]^T\,. \tag{4.4}$$

The least squares estimates are found analytically by setting to zero the partial derivatives of SSE with respect to a, b, and c and by solving the resulting normal equations. However, the normal equations for the aforementioned exponential model are nonlinear in the parameters, so no closed-form solution exists. Consequently, such a solution is hard to find. The complexity of the problem is further increased when multiple solutions exist. An easier alternative is to search for the values of γ^* by trying various values of a, b, and c and evaluating SSE until a minimum is found.

In the next section the parameters of the model for the test circuits are determined experimentally. This model will be used as the underlying framework for understanding and evaluating the solution quality later on.

4.2. Justification of the Exponential Model

In this section we present the results to support the claim of exponential behavior of the incumbent. The two benchmarks chosen for the experiments are the PrimGA1 and PrimGA2, with 752 and 2900 cells respectively. These are the suggested benchmarks for gate array layouts. In addition, we also present results for two other circuits, EX1 and EX2, with 200 and 1000 cells respectively. These four circuits help to demonstrate the validity of the model we propose in this paper. The experiments have been carried out on a Sun SPARC workstation.

Two sets of experiments have been carried out: one for the static case and the other for the dynamic case. In the static case the entire trace of the incumbent values is assumed available beforehand for fitting the model. Though not a realistic scenario, the static case helps in establishing the accuracy of the dynamic predictions. In the dynamic case, the parameters of the proposed model are initially determined from a small set of incumbent values, and these parameters are refined as more incumbent values become available. The dynamic case is thus an incremental process.

The behavior of the incumbent for two benchmarks is shown in Figure 2. The plots in this figure have been normal-

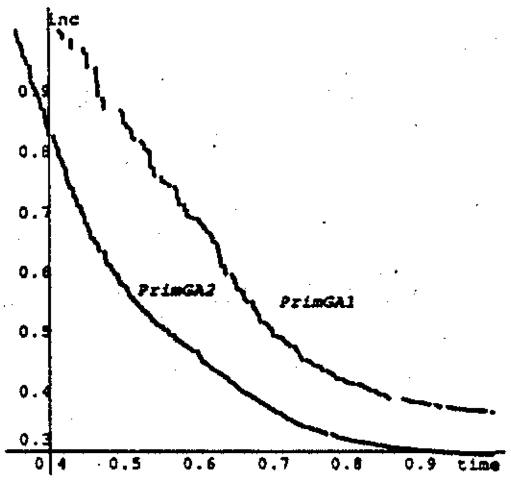


Figure 2. Behavior of incumbent with time

ized. The simulated annealing algorithm has been run to completion following a fixed annealing schedule.

There are two aspects involved in exponential curve fitting. The first is to determine the constant c in the exponential model. This constant defines the final incumbent value were the experiment to be continued for infinite time. This, in effect, would be the optimal solution of the placement task under the assumption that the exponential curve is an exact fit for the incumbent. In practice, this is not the case since it is not possible to explore the entire configuration space of a large problem in a reasonable amount of time; hence, there will be some errors in the fitted curve. The second part of the curve fitting is on determining the decay constants a and b. These two constants determine the rate of decrease of the incumbents with time.

The next task is to find suitable values for a, b and c so that SSE is minimized. This task would be greatly simplified if it could be established that SSE is a convex function of c. In the latter case a binary search can be performed to determine the appropriate value of c. Attempts to establish the convexity of SSE with respect to c are rendered difficult owing to the nonlinearity of the expressions involved. We show that SSE is not convex by finding counter examples. This is achieved as follows. We start with an exponential curve of the form $(c + exp(-at+b))$. Points lying on this curve are treated as the observed incumbent values, implying that the curve is a perfect fit for the incumbent. The value of c is then varied, and using the aforementioned incumbent values sampled at discrete time points a new fit is determined. Since c is varied, the values of a and b for the new fit are different from those of the original curve. The sum of squares error, SSE, is then found. The result plotted in Figure 3a shows that the relationship between SSE and c is not convex. Next, we consider the actual incumbent values collected during the simulated annealing process. The exponential fits for different values of c are then determined. We find that the plot of SSE versus c has more than one local minima, as seen in Figure 3b. A possible reason for this is that data points that do not lie on the fitted curve and are near the

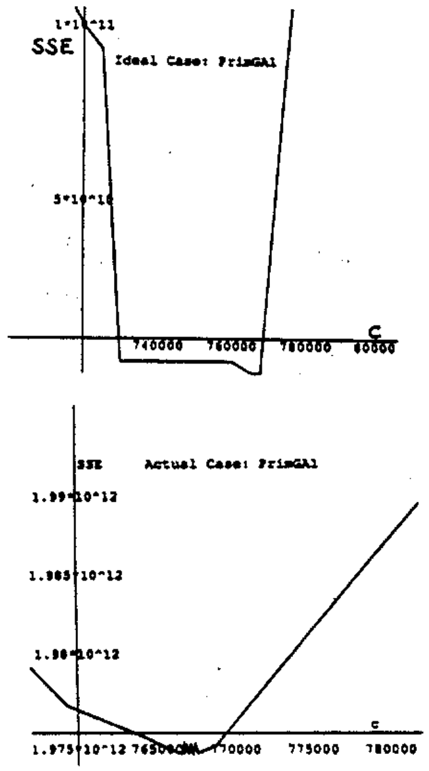


Figure 3. Variation of SSE with respect to c for fitting (a) points generated from an exponential curve and (b) incumbent values collected in simulated annealing.

extremities dominate the value of SSE. A binary search for possible c values in this case essentially leads to some local minimum. It is worthwhile to mention here that the model suggested in the paper accords a statistical relationship and not a functional relationship between the incumbents and the time. Hence, not all data points will lie on the fitted curve.

The maximum value that c can assume is determined by the value of the incumbent at the time the annealing experiment is stopped. This is obvious as the incumbent decreases monotonically. The other extremity for c is determined heuristically. The value of c that results in the smallest value for SSE in the chosen range is selected. Due to the heuristic nature of this method, there is an error associated with the value of c found this way. However, this is the best that can be done given the limited knowledge of the configuration space of the placement problem. *Mathematica*, a package for performing mathematical computations, was used to find the parameters a and b for the exponential fit.

In the next section we present our results and demonstrate how the predictions on the incumbent values can be used in trading between solution quality and time.

4.3. Experimental Results on Tradeoffs

The descriptions in this section deal with the tradeoff of solution quality and time spent. Different yardsticks can be used to estimate this tradeoff depending on the emphasis desired. Consequently, various objective functions, also termed heuristics, can be stated. The following theorem states the conditions to be met by a heuristic for this purpose.

Theorem 1. If $f(t)$ is the incumbent and $g(t)$ is a function of time, both being piecewise continuous and twice differentiable on R^1, then the product $f(t)g(t)$ is minimized with respect to time if, for $t \in R^1$,

$$f''(t)g(t) + 2f'(t)g'(t) + f(t)g''(t) \geq 0 \qquad (4.5a)$$

$$\text{and } f'(t)g(t) + f(t)g'(t) = 0 \qquad (4.5b)$$

has a solution in R^1.

Proof. The theorem establishes the condition for convexity and the proof follows naturally. □

In this paper we have explored one such heuristic. It is defined as the quality-time ratio or incumbent-time product. Note that quality has been defined earlier as the reciprocal of the incumbent value. The functions are

$$f(t) = c + e^{(-at+b)}, \quad \text{and} \quad g(t) = t, \qquad (4.6)$$

and the heuristic is

$$h(t) = c\,t + t\,e^{(-at+b)}. \qquad (4.7)$$

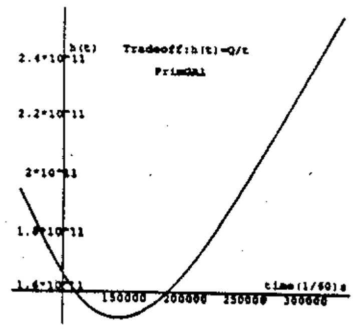
We wish to study how this heuristic varies with time. The aim is to maximize the ratio Q/t, or in other words minimize the product of incumbent value and time. This objective function places equal emphasis on solution quality and time. Determination of the time at which the heuristic is optimized gives an indication of the point at which the algorithm can be stopped. In this particular case, the condition to be met in accordance with the theorem stated above is

$$t > \frac{2}{a} \qquad (4.8)$$

This heuristic has been studied to demonstrate the usefulness of the model proposed in this paper. The following experiment has been carried out using the same data that was collected for the experiments in Section 4.1. Figure 4 shows the plot of $h(t)$ versus time for the static case for the two benchmarks, PrimGA1 and PrimGA2. The values of incumbents for this plot are obtained from the exponential fit determined earlier.

From the figures the convexity of the function h(t) is evident as is the time at which the minimum occurs, denoted in the paper by t_{static}. After t_{static}, the solution does not improve substantially as more time is spent. Therefore, it is safe to conclude that the annealing algorithm can be terminated at t_{static}, since there is no substantial improvement in solution quality if the experiments were continued.

However, all of the above is for the *static* case in which it is assumed that the simulated annealing algorithm has been run to completion according to a prespecified annealing schedule. Thus all the values of the incumbent are available, and it is easy to fit a curve to it. The experiment was carried out mainly to demonstrate the feasibility of the method and lay the groundwork for the dynamic case. In practice, however, we wish to be able to predict the nature of the incumbent curve with just a few values collected at the beginning of the experiment. We refer to this as the *dynamic* case. Dynamic prediction of the incumbent would help in terminating the annealing algorithm should time be a constraint. We propose to do this extrapolation in an incremental fashion as described below.



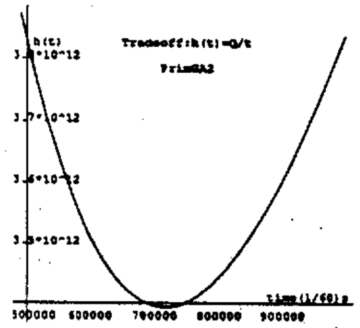


Figure 4. Quality-Time Tradeoff Curves (Static Case) for benchmarks (i) PrimGA1 and (ii) PrimGA2 .

The annealing algorithm is run for a fixed number of iterations, during which the incumbent values are recorded. An exponential curve is fit to the recorded values along lines described for the static case. The annealing algorithm is then continued for the next predetermined set of iterations and the incumbent values recorded as before. The exponential fit is then modified to suit the additional set of values for the incumbent. This process is continued until the annealing algorithm is stopped in accordance with the prespecified time constraint. Using the latest exponential fit, the function h(t) is evaluated and the time at which the minimum occurs, t_{pred}, is determined.

The results obtained are shown in Tables 1 and 2. The notations used are as follows: t_{static} refers to the stopping time predicted in the static case; t_{pred} refers to the stopping time predicted by the dynamic method for the samples collected so far; $t_{dynamic}$ is the actual stopping time of the dynamic method; t_{sample} refers to the time at which the sample was collected; and t_{comp} refers to the time required by the annealing algorithm to complete the run as per the annealing schedule. The notations used for the incumbent values follow from the above terminology. It is to be noted here that inc_{comp} is the best incumbent value that is achieved by the annealing algorithm.

Table 1 shows t_{sample}/t_{static}, the normalized times incurred so far in the process, and t_{pred}/t_{static}, the normalized times predicted at which the process should stop. Hence, the annealing process can be terminated when the normalized predicted time is less than or equal to the normalized time incurred. This condition is satisfied in the last entry of each column in Table 1. Table 1 also shows the error incurred in the dynamic method. The overshoot in the simulated annealing algorithm using dynamic prediction is indicated by $(t_{sample}/t_{static} - 1)$: errors for all four circuits range between 7% and 11%. The entries marked "-" indicate that a minimum could not be found for the heuristic using the samples available. This occurs in the early stages of the process when the samples are not reflective enough of the trend. It conveys the fact that more samples need to be collected for the prediction to be meaningful.

An evaluation of the effectiveness of the quality-time tradeoff is presented in Table 2. The figures obtained for the static and dynamic cases are indicated.

Table 1. Time incurred vs. predicted stopping time.

EX1		EX2		PrimGA1		PrimGA2	
$\frac{t_{sample}}{t_{static}}$	$\frac{t_{pred}}{t_{static}}$	$\frac{t_{sample}}{t_{static}}$	$\frac{t_{pred}}{t_{static}}$	$\frac{t_{sample}}{t_{static}}$	$\frac{t_{pred}}{t_{static}}$	$\frac{t_{sample}}{t_{static}}$	$\frac{t_{pred}}{t_{static}}$
0.92	6.88	0.80	-	0.73	-	0.80	7.99
0.96	1.63	0.95	-	0.78	35.12	0.88	-
0.99	1.2	0.98	10.36	0.82	15.04	0.94	1.15
1.06	1.09	1.01	1.84	0.87	15.05	1.07	1.03
1.09	1.04	1.05	1.13	0.93	2.08		
		1.09	1.11	1.00	1.27		
		1.11	1.11	1.05	1.07		
				1.07	1.07		

Table 2. Summary of Quality-time Tradeoffs

Circuit	Static		Dynamic	
	$\frac{t_{static}}{t_{comp}}$	$\frac{inc_{comp}}{inc_{static}}$	$\frac{t_{dynamic}}{t_{comp}}$	$\frac{inc_{comp}}{inc_{dynamic}}$
EX1	0.82	0.88	0.90	0.95
EX2	0.55	0.88	0.61	0.94
PrimGA1	0.92	0.97	0.99	1.0
PrimGA2	0.72	0.83	0.77	0.89

The table affords two useful evaluation criteria. As an example, consider the test circuit EX1: the static case indicates that 82% of the time was spent to achieve a solution accuracy of 88% (accuracy being relative to inc_{comp}). This is the best tradeoff that can be obtained since in static prediction all sample points are assumed available for determining the best exponential fit. A close examination at the corresponding dynamic case suggests that by spending 90% of the time in annealing, a solution of accuracy 95% is achieved. This means that an additional 10% of the time would fetch a solution which is only 5% better. It is clear then that depending upon the discretion of the circuit designer, such a figure of merit can be used to achieve the desired tradeoff.

The time required for performing the dynamic tradeoff is a small fraction of the overall time for annealing. The overhead incurred varies from 9% for the benchmark PrimGA2 (2900 cells), to 25% for the benchmark EX1 (200 cells). This is understandable, since for small-sized circuits annealing requires less time to complete and performing a tradeoff is not so meaningful. In a similar vein, for larger circuits annealing requires a lot of time to complete and so tradeoff estimation is a small fraction of the annealing time. The fact that tradeoff is useful for large circuits has already been emphasized earlier in the paper. In all the cases curve fitting is carried out seven times during the course of annealing, which is a quite high. The percentage of time quoted above would obviously be much smaller when the frequency of curve fitting is low. It is important to note here that the tradeoff estimation can go either in tandem with the annealing process or in parallel depending upon the resources available.

5. Conclusions

In this paper, we have discussed a method that examines the tradeoff between the quality of the best solution, incumbent, and the computational time of a simulated annealing process for VLSI placement. An exponential model for the incumbent behavior has been proposed and validated by experiments. The results indicate that dynamic predictions provide a good estimate of the tradeoff achieved in the annealing algorithm and could be very useful in practice to reduce the design cycle time. The method proposed can be extended to other applications that rely on iterative algorithms. Such a study would be useful in cases in which two or more heuristic methods are available, and a combination of these methods needs to be used to generate a good solution to the problem. The method described in this paper can be used to schedule these heuristic methods on the basis of their performance at any time. Future work includes a study of dynamic prediction for standard cell layouts and the scheduling of different heuristics for the placement problem.

REFERENCES

[1] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization By Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671-680, American Association for Advancement of Science, Washington, D.C., May 1983.

[2] S. Sahni and A. Bhatt, "The Complexity of Design Automation Problems," *Proc. 17th Design Automation Conference*, pp. 402-411, IEEE/ACM, 1980.

[3] C. Sechen and A. S. Vincentelli, "TimberWolfe3.2 : A New Standard Cell Placement and Global Routing Package," *Proc. 23rd Design Automation Conference*, pp. 432-439, IEEE/ACM, 1986.

[4] C. Sechen, *VLSI Placement and Global Routing Using Simulated Annealing*, Kluwer Academic Publishers, Boston, MA, 1988.