A Force Directed Hill-Climbing Placement Algorithm

R. Mall
Dept. of Computer Science and Automation
Indian Institute of Science
Bangalore 560 012

L.M. Patnaik
Microprocessor Applications Laboratory
Indian Institute of Science
Bangalore 560 012

e-mail: lalit@vigyan.ernet.in

Abstract

In this paper, we propose a heuristic procedure for finding near-optimal placement of VLSI circuit modules on a two-dimensional layout. This heuristic procedure combines ideas from a greedy placement algorithm and the Probabilistic Hill-Climbing technique. The main advantage of this alpproach is that while it retains the fast convergence property of the greedy algorithm, it exhibits adaptive hill-climbing capability. Performance evaluation studies on this algorithm have been carried out and the results appear promising. The results indicate that while the algorithm is computationally less expensive than the simulated annealing algorithm, it yields better final placement solutions than the greedy algorithm.

1. Introduction

The general placement problem of VLSI circuit layout is that of placing a set of circuit modules on a layout area such that a set of objective functions is optimized. An important objective is the minimization of the total interconnection length. The problem of finding optimal module placements subject to minimization of the objective functions is known to be an NP-hard problem [7]. For solving such computationally difficult problems, algorithms based on *greedy* heuristics yielding suboptimal solutions are usually employed in practice.

Greedy algorithms typically work in two phases. In the first phase, an initial suboptimal solution is proposed. In the second phase, the initial solution is improved through a series of perturbations and greedy acceptances. However, greedy heuristics get stuck at local optima. Though this is a major bottleneck, greedy algorithms are still very popular, primarily due to the fact that they yield "good" solutions while being computationally inexpensive.

Probabilistic Hill-climbing (PHC) algorithms have been suggested for obtaining optimal solutions [6] to difficult optimization problems. The intuitive idea behind this approach is to generate new states via perturbations and to accept all the states giving better configurations. Also, some of the generated states giving worse configurations, are accepted based on certain probability distribution. This acceptance of "worse states" is known as the hill-climbing steps. It has been proved that PHC algorithms can converge to globally optimum solutions[1]. However, the main disadvantages of the PHC algorithms are that they require enormous amounts of computing time [8], and that they are not amenable to efficient parallel implementation [5]. Consequently, there is a need for developing fast algorithms that can give results of quality comparable to those of the PHC algorithms.

In this paper, we propose a hybrid algorithm that is, in essence, a modification of the heuristic procedure reported in [2] by applying ideas from the PHC technique. We have named this hybrid algorithm Generalized Force Directed Relaxation with Probabilistic Hill-Climbing (GFDR-PHC) algorithm. GFDR-PHC has the characteristic of a greedy algorithm possessing "some" hill-climbing capability. The main motivation behind the development of this algorithm is to retain the advantage of fast convergence of greedy algorithms while at the same time realizing near-optimal solutions. GFDR-PHC algorithm has been coded and its performance has been studied. The results appear promising.

The rest of this paper is organized as follows. In the next section we describe some preliminary concepts and definitions. Section 3 presents the GFDR-PHC algorithm. Section 4 gives comparative performance analysis of the algorithm. Section 5 presents the performance evaluation studies. Section 6 concludes this paper.

2. Preliminary Concepts

The greedy part of the GFDR-PHC algorithm is based on the Generalized Force Directed Relaxation (GFDR) algorithm reported by Goto [2]. The algorithm proposed in [2] is a popular algorithm that is widely used and referred to by researchers. This algorithm operates in two phases [2]. The first phase called SORG (Sub-Optimum Random Generation), generates an initial constructive placement. The second phase of the algorithm GFDR (Generalized Force Directed Relaxation), carries out iterative improvements to the initial placement. Details of this algorithm can be found in [2]. We give a few important definnitions below.

Definition 2.1

The **wiring length** (or **routing length**) of a signal net is the half-perimeter of the smallest rectangle enclosing all modules (terminals) of the net.

Definition 2.2

The total wiring length of all signal nets associated with a module M is denoted by **MWL(M)**.

Definition 2.3

The **median of a module** is the position of a module M, where the total wiring length associated with the module (i.e., MWL(M)) is minimum.

Definition 2.4

The **ε-neighborhood** of a module M is the set of ε locations where the total wiring length associated with this module is minimum (i.e., first ε minimum MWL(M) locations). The **ε-neighborhood** of a module M is obtained by

0-8186-2465-5/91 $1.00 © 1991 IEEE

first sorting the values of MWL(M) for different positions of M are in ascending order. The first ε elements of this list give the **ε-neighborhood** for the module.

2.1. The GFDR Algorithm

The GFDR algorithm operates by examining a search tree constructed by recursively interchanging a module with the modules in its ε-neighborhood. The GFDR algorithm operates for a fixed value of the ε-neighborhood and for a fixed value of the maximum number of modules involved in any interchange sequence (λ^*), i.e., fixed depth of the search tree. The number of modules involved in a module interchange sequence is called the level of interchange. First, the algorithm randomly selects a module (level = 1). The level 1 module is called the *primary* module. Pairwise trial-interchange of this module with each of the modules in its ε-neighborhood is carried out (level = 2). The best interchange is accepted, if it results in a cost reduction. If no cost reduction for this module takes place in level = 2, then level = 3 (three module interchange sequence) is tried out and so on. This procedure continues till level λ^* (maximum interchange level) is reached. This procedure is repeated for different primary modules.

3. GFDR-PHC Algorithm

GFDR-PHC algorithm retains the constructive placement part (SORG) of the algorithm described in [2] and modifies only the iterative improvement (GFDR) part of the algorithm. In order to appreciate the modifications to the GFDR algorithm, let us regard the modules in the initial placement configuration to be joined together by lines of force (similar to springs) that are under tension, as in Hanan [3]. Our primary objective is to minimize the *energy* (wire length) of the initial placement, subject to the constraint that each module is assigned to a unique slot on the chip. In the sequel, we will use the term *energy content of a module* for its total module wiring length and vice versa. The terms *median position* and *minimum energy location*, are used interchangeably. Also, the terms *energy release* and *reduction in wiring length* are used with the same implication.

When a module is relaxed (allowed to move freely), it moves to a minimum energy location. This minimum energy position is analogous to the median location of GFDR algorithm. Let us consider an ε-neighborhood of this module, this contains the first ε *minimum energy locations*. When a module under tension is relaxed and is positioned at one of its ε-neighborhood locations, the system goes to a lower energy state, that is, the total wiring length decreases. In the first phase of the algorithm, an initial constructive placement takes place. In the second phase of the algorithm, a module M (from an abitrary location location p) is selected. This module is called as the primary module. The primary module is trial interchanged with each of its ε-neighborhood modules, e.g., in Fig.1, module M is trial-interchanged in turn with modules X, Y, and Z respectively, where X, Y, and Z are the ε-neighborhood of module M. The total change in the wiring length = MWL(M,P) - MWL(M,Q) + MWL(J,Q) - MWL(J,P), for each trial-interchange of module M located at P with a module J $\in$ {X,Y,Z} located at Q is noted. The trial-interchange which gives the maximum decrease in wiring length is noted and the corresponding routing length as a result of this trial-interchange is remembered as the minimum routing length uptill now. The module at the ε-neighborhood location giving this minimum value becomes the current module and is chosen for the next level of interchange. The chain from the primary module to the current module (M-Y in Fig.1) giving the maximum decrease in routing length is stored in a linked list for future reference.

In the next step, the current module (Y) is relaxed (second level relaxation), and placed at each of its ε-neighborhood locations. Round robin interchange starting with the primary module and ending up with each one of the ε-neighborhood locations of the current module is effected (M-Y-A, M-Y-B, and M-Y-C in Fig. 2). For each such interchange sequence, the change in wiring length is noted down. The module in the ε-neighborhood location, that gives the maximum decrease in wiring length is selected as the current module. If the change in wiring length in this case results in a value of routing length less than the earlier minimum routing length, then the latter is updated and the minimum chain is extended upto the current module. Even though a particular level of relaxation may not result in an immediate reduction in minimum routing length, the relaxation sequence continues provided the total routing length decreases. Thus, the change in MWL of a module at any single level of relaxation is allowed to be either positive or negative, as long as the *total* routing length decreases.This is analogous to a hill climbing step, because inspite of a "bad move" the search process continues.

The process of examining the ε-neighborhood locations of the current module is continued till the level of interchange where all ε-neighborhood locations give increases in total wiring length. The round robin interchange of modules corresponding to the minimum chain giving the best interchange sequence (i.e. resulting in minimum total routing length) is carried out. The current routing length is updated as the minimum routing length for the next iteration. In an interchange sequence, no module is allowed to be relaxed more than once as otherwise it may result in improper cost calculations. In one cycle of operation, every module is selected once as the primary module. The algorithm terminates, when a complete cycle of relaxation does not produce any improvement to the placement configuration.

A major advantage of the GFDR-PHC algorithm is that the search procedure does not have a fixed λ^* value, but continues as long as the change in the *total wiring length* is negative. Thus, it does not stop just because negative cost change occur at some particular interchange. GFDR-PHC algorithm is outlined below.

Algorithm GFDR-PHC;
/* A cycle of the algorithm selects each module once as the primary module. Initially, current-routing-length = total-routing-length obtained by SORG */
step 0: chain = NULL;
/* chain stores module interchange sequences */

```
step 1: M = next primary module;
step 2: current-module = M;
step 3: /* initialization */
   3.1: min-routing-length = current-routing-length;
   3.2: least-wirelength-change = 0; /* initialization */
   3.3: append(chain, current-module);
                /*store current module in chain */
step 4:ε-neighbor = find_ε-neighbor(current-module);
  /* determine ε-neighborhood of current-module */
step 5: for each module i ∈ ε-neighbor do
   5.1: round_robin_interchange(i,chain);
/* interchange all modules from primary module
upto module i */
   5.2:wirelength-change = Σ (MWL(j,p) − MWL(j,q));
/* Each module in the chain is trial interchanged
with the next module, and the new wire length after
the entire intechange sequence is found out. */
   5.3: if(wirelegth-change < least-wirelength-
change)then
   5.3.1:least-wirelength-change = wirelength-change;
   5.3.2:least-module = current-module;
        /* remember current-module */
   5.4: reverse-interchange(i,chain);
    /* restore the original configuration */
step 6: current-module = least-module;
step 7:if((current-routing-length + least-wirelength-
change) < min-routing-length) then
   7.1: append(chain,current-module);
    /* store current-module in chain*/
   7.2: min-routing-length = current-routing-
length + least-wirelength-change;
        /* update value of min-routing-length */
step 8: if (least-wirelength-change ≥ 0)
  then goto step 10 ; /*no improvement possible */
   else goto step 3; /*start next level search */
step 9: round-robin-interchange(chain);
 /* carry out interchange giving minimum total
wiring length */
step 10: current-routing-length = min-routing-length;
step 11: if cycle not over then goto step 0;
/* in one cycle each module is chosen once as
primary module */
step 12: if(improvement over previous cycle) goto
            step 0 else stop.
```

4. Analysis

A closed-form analysis of the performance of the algorithm becomes complicated due to the hill climbing steps involved. However, an intuitive analysis of the algorithm is given below.

The GFDR-PHC algorithm is develped based on two simple observations. The first one is that, maximum energy reduction in a module interchange sequence occurs when a the ε-neighborhood of a relaxed module contains the primary module. This is due to the fact that the relaxed module moves into the slot of the primary module, thus forming a cycle. The probability of finding such an interchange sequence becomes greater for larger values of the depth of interchange. Thus, when large sequences of module interchanges are allowed to occur, there is higher chances of finding interchange sequences of the type just described. Secondly, in a PHC algorithm better solutions (crystalline order, in statistical mechanics terms) are obtained when more uphill climbs are allowed to take place during the hot phase in the cooling schedule than during the cold phase; in the reverse case locally optimum (glassy) solution are likely [4]. Simulated Annealing algorithm achieves this by decreasing the acceptance probability of a move as the temperature reduces. GFDR-PHC algorithm inherently incorporates this principle. Initially, the layout is at a high energy state as the non-optimally placed modules are under great tension. Each time a module is taken up for relaxation, large reductions are achieved. In this phase, many interchange sequences are carried out which do not immediately decrease the MWL because of the large positive values of the total wirelength change. Thus large number of uphill climbing steps take place during this phase. However, when the system cools down, that is, the modules are placed near-optimally and the relaxation steps do not release much energy, less uphill climbs can be entertained since the total wirelength change usually has small positive values. The initial temperature and the cooling schedule are automatically taken care of by the algorithm (i.e., these are adaptive) as was suggested by Nahar [5], being highly dependent on the initial placement configuration.

With this interpretation, we can realize that in the original GFDR algorithm, uphill climbs take place in the reverse sequence. Since, initially modules are suboptimally placed, almost every interchange sequence results in decrease in total wirelength. Hence, according to the GFDRin the initial phase few uphill moves are tried (search terminates mostly at level = 2). Whereas, later when the system cools down, more and more uphill moves are attempted (search mostly continues up to λ^* level). Another important fact is that in the original GFDR algorithm, the value of λ^* (maximum allowed interchange level) has to be fixed at the start of the algorithm. But in GFDR-PHC algorithm, λ^* can take any value from 2 upwards, depending on optimality of the placement of the modules under consideration. Thus, λ^* has been made adaptive.

5. Performance Studies

The GFDR-PHC algorithm has been implemented on a SUN 3/260 Workstation. Also, for the sake of performance comparison, the GFDR algorithm and the SA algorithm have been implemented on the same machine. The relative performances of these three algorithms has been determined by running each of them for the same set of test problems. The summary of performance results is presented in Figs. 3 and 4. In the implementation of both GFDR and GFDR-PHC algorithms, the optimal value of ε has been taken to be 3 as suggested in [2].

From Fig. 3, we can observe that the solution given by the GFDR-PHC algorithm is better than that given by the GFDR algorithm and closer to the solution given by the simulated annealing algorithm. Fig. 4 reflects the fact that

GFDR algorithm has almost a linear performance with respect to the number of nets. Fig. 4 also shows that GFDR-PHC algorithm outperforms the SA algorithm in computation time requirements, although it takes relatively more time than the GFDR algorithm for identical problems. The results are in conformance with the discussions of sections 4.

6. Conclusions

In this paper, we have proposed a hybrid placement algorithm that retains the fast convergence property of a greedy algorithm, while at the same time yields near-optimal solutions using probabilistic hill-climbing steps. Experimental studies to determine the performance of the algorithm have been carried out, the results agree with the predicted performance of the algorithm. Our current work is directed towards exploring the possibility of applying the presented idea to other greedy algorithms to improve the quality of their final solutions.

References

[1] E.H.L. Aarts *et al.*, "Statistical Cooling: A general approach to combinatorial optimization problems", Philips Journal of Research, Vol. 40, No. 4,1985, pp. 193 - 226.

[2] S. Goto, "An efficient algorithm for the two dimensional placement problem in electrical circuit layout", IEEE Transactions on Circuits and Systems, Vol. CAS-28, Jan 1981, pp. 12 - 18.

[3] M. Hanan and J.M. Kurtzberg, "Placement techniques", in *Design Automation of Digital Systems Vol. 1: Theory and Techniques*, Ed. M.A. Breuer, Eaglewood Cliffs, New Jersey, Prentice Hall, 1972, pp. 213 - 282.

[4] S. Kirkpatrick, C.D. Gelatt Jr., and M.P. Vecchi, "Optimization by simulated annealing", Science, 13, May 1983, Vol. 220, No. 4598, pp. 671 - 680.

[5] S. Nahar *et al.*, "Simulated annealing and combinatorial optimization", in Proc. 23rd Design Automation Conference, pp. 29-297, 1986.

[6] F. Romeo *et al.*, "Research on simulated annealing at Berkeley", Proc. Int. Conf. on Comp. Design, pp. 652- 657, Oct. 1984.

[7] S. Sahni and A. Bhatt, "The complexity of design automation problems", in Proc. 17th Design Automation Conference, pp. 402-411, 1980.

[8] S. Sechen and A. Sangiovanni Vincentelli, "The Timberwolf placement and routing package", in Proceedings of the 21st Design Automation Conference, pp. 522 - 527, 1984.

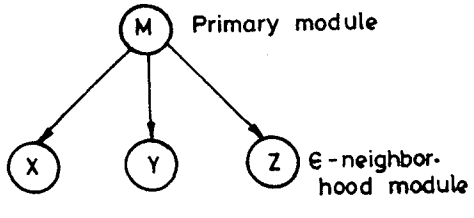

FIG. 1 LEVEL-2 SEARCH TREE

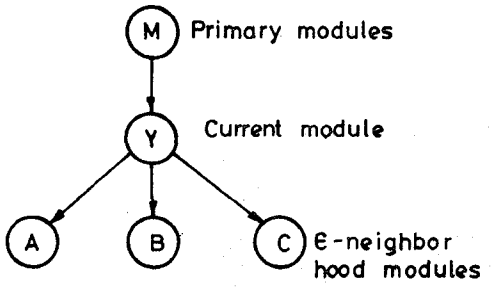

FIG. 2 LEVEL-3 SEARCH TREE

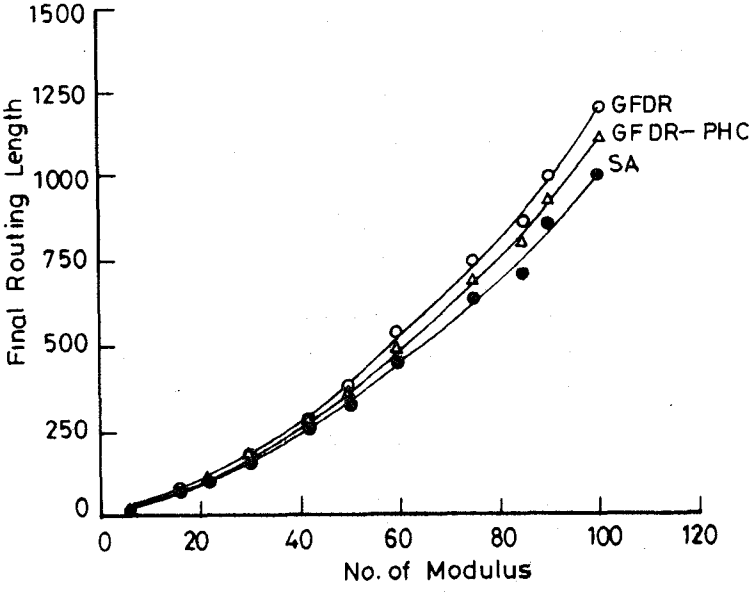

FIG. 3 PERFORMANC OF THE ALGORITHMS W.R.T ROUTING LENGTH

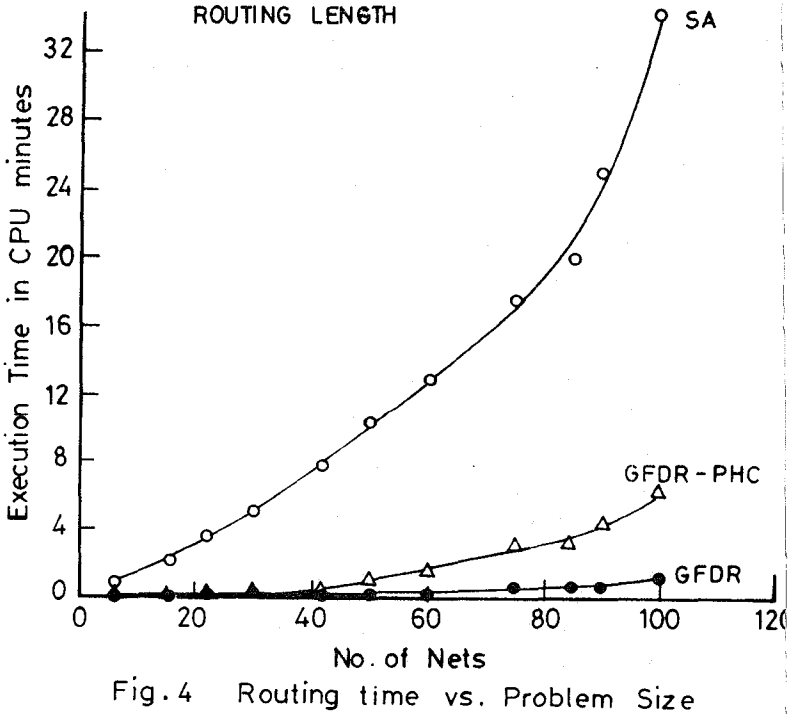

Fig. 4 Routing time vs. Problem Size