CIRCUIT PARTITIONING USING A TABU SEARCH APPROACH

Shawki Areibi*, Anthony Vannelli †
University of Waterloo
Department of Electrical and Computer Engineering
Waterloo, Ontario CANADA N2L 2G1

ABSTRACT

Tabu Search is a simple combinatorial optimisation strategy that has been applied with great success in applications ranging from graph coloring to scheduling and space planning. This paper describes the application of the Tabu Search[1] heuristic to the circuit partitioning problem. Results obtained indicate that in most cases Tabu Search yields netlist partitions with 10% fewer cut nets than the best netlist partitions obtained by using an interchange method or Simulated Annealing. Moreover, the Tabu Search method is 3 to 20 times faster than Simulated Annealing on tested problems. This paper also describes the benefit of integrating Tabu Search with Simulated Annealing.

1 INTRODUCTION

Circuit partitioning is the task of dividing a circuit into two or more disjoint blocks in such a way as to minimise the number of connections between the blocks of the partition. Partitioning is an important aspect of layout for several reasons. Partitioning can be used directly to divide a circuit into portions that are implemented on separate components, such as printed circuit boards or chips. Here the objective is to partition the circuit into parts such that the sizes of the components are within prescribed ranges and the complexity of connections between the components is minimised.

It has been shown that graph and network partitioning problems are NP-Complete[2]. Therefore, attempts to solve these problems have concentrated on finding heuristics which will yield approximate solutions in polynomial time. For the circuit partitioning problem three different classes of algorithms are used to generate good partitions. These techniques are, *Iterative Improvement algorithms* [3], *Numerical Optimization Techniques* [4] and *Simulated Annealing* [5].

*sareibi@cheetah.vlsi.uwaterloo.ca
†vannelli@cheetah.vlsi.uwaterloo.ca

0-7803-1254-6/93$03.00 © 1993 IEEE

2 TABU SEARCH

Originally proposed in 1977 by Glover [1] as an optimisation tool to solve nonlinear covering problems, Tabu Search has recently been applied to problems such as integer programming, scheduling and graph coloring.

2.1 A Simple Form of Tabu Search

We first present Tabu Search in a simple form that discloses two of its key elements: that of constraining the search by classifying certain of its moves as forbidden (i.e., Tabu) and that of freeing the search by a short term memory function that provides "strategic forgetting". Figure 1 provides an example of the Tabu Search mechanism. Figure 1a shows a circuit that is to be partitioned into two blocks with an initial random partition. Figure 1b presents the gains associated with modules in each iteration (always pick the module associated with best gain "steepest descent") Figures 1d,e give the status of the list of Tabu solutions as the routine progresses. Note that moves do not necessarily result in a decrease in the associated cost; costs may increase simply as a result of the current solution being Tabu.

For the purpose of Figure 1c, only four consecutive solutions are considered as Tabu at any one time. Starting from an initial solution s_0 a move is made to solution s_1 on the basis of cost. At the same time, the initial solution s_0[1] is placed in a list of Tabu solutions, which is, in effect, a first-in-first-out queue of length 4. On the basis of accepting the allowable (i.e., non-Tabu) solution with the minimum cost, this process is repeated in subsequent moves. Note that, in the move from s_4 *to* s_5, the Tabu status of the initial feasible solution s_0 is dropped; whenever a fifth solution enters the Tabu list, the oldest solution therein is removed and again becomes allowable. The procedure described so far lacks one important feature suggested by Glover [1]. This feature is known as Aspiration and is described in the following subsection.

[1] In this case the movement of module 5 from block 0 to block 1 is s_0.

Authorized licensed use limited to: University of Johannesburg. Downloaded on March 05,2010 at 03:02:43 EST from IEEE Xplore. Restrictions apply.

2.2 Aspiration Level

Aspiration is a heuristic rule based on cost values that can temporarily release a solution from its Tabu status. The purpose of Aspiration is to increase the flexibility of the algorithm while preserving the basic features that allow the algorithm to escape local optima and avoid cyclic behavior. At all times during the search procedure each cost c has an Aspiration value that is the current lowest cost of all solution arrived at from solutions with cost c. The actual Aspiration rule is that, if the cost associated with a Tabu solution is less than the Aspiration value associated with the cost of the current solution, then the Tabu status of the Tabu solution is *temporarily* ignored.

Figure 1c, together with the table in Figure 1f, illustrate the advantageous effects of Aspiration. Initially, at step 0, all Aspiration values are set at a level higher than any possible cost indicated in Figure 1f as "hi". As moves are made, the appropriate Aspiration values are updated, if necessary. For instance, when the first move is made from a solution of cost 8 to a solution of cost 6, the Aspiration value associated with the cost value of 8 is updated in Figure 1f to 6. This implies that, for Aspiration to release a Tabu solution so that it may become a candidate for a move from any solution with cost 8, the Tabu solutions must have a cost less than 6. Similarly, the second move forces an update of the Aspiration value associated with a cost value of 6; the Aspiration value is now set at 7.

In the example given in Figure 1e, the use of Aspiration allows the important seventh move back to a previously accepted solution. In this move the Aspiration value associated with solution (cut 6 s_6) is 7 as seen in Figure 1e, this value is greater than the solution (cut 5 s_7) which is already Tabu (in the Tabu List). According to the Aspiration rule, the Tabu status of this Tabu solution is temporarily ignored and the move is allowed, hopefully leading in the direction of the optimal solution of cost one. The Tabu Search routine described in detail in the previous subsections can be formulated as shown in Figure 2.

3 INTEGRATING TABU SEARCH INTO SIMULATED ANNEALING

The discussion of Tabu Search in the previous sections suggest that Tabu Search may be considered as a kind of deterministic version of Simulated Annealing in a broad sense. It is evident that stochastic, adaptive and local search approaches have strengths and weaknesses and that they should be viewed not as competing models but as complementary ones. By integrating these fundamentally different approaches we can avoid many of the weaknesses inherent in each methodology, while capitalising on their individual strengths. In addition, most real world problems especially circuit layout are too complex for any single processing technique to solve in isolation.

Simulated Annealing achieves diversity in search by randomisation without reliance on memory. By this view, the use of randomisation, via assigned probabilities, allows a gain in efficiency by obviating extensive record keeping and evaluation operations that a more systematic pursuit of diversity may require. At the same time, it entails a loss in efficiency by allowing duplications and potentially unproductive wandering that a more systematic approach would seek to eliminate. One possible effort to improve Simulated Annealing is to replace its rules with those more closely resembling the prescriptions of Tabu Search. We show that the combination of Tabu Search and Simulated Annealing gives rise to probabilistic (hybrid) algorithms that form the basis for approaching combinatorial optimisation problems in a semi-intelligent manner.

4 TESTS AND RESULTS

This section compares the results obtained by the Tabu Search method with those obtained by Simulated Annealing [5] and Sanchis multiple-way partitioning interchange method [3]. These techniques and the resulting computer codes were tested on four netlist partitioning problems as seen in Table (1).

All computer tests were made on SPARC-based computers (either SPARC II or IPC) running SunOS version 4.1. The programs were written in C and compiled with the Sun C compiler. We should note that the *cooling schedule* used with Simulated Annealing is similar to the simple cooling schedule proposed by Kirkpatrick [5]. Due to this approximation, the algorithm is no longer guaranteed to find a global minimum with probability one.

Some general observations can be made from the results presented in Tables (2) and (3). The total of the execution times obtaining a final partition for the different methods indicate that the interchange technique takes the least amount of CPU time whereas the Simulated Annealing approach takes quite a long CPU time. Tabu Search execution time is faster than the Simulated Annealing approach by a factor of 10 and is competitive with the interchange method. The quality of partitions obtained by the Tabu Search method are better than those obtained by the Iterative method by 20% and yields partitions that are 10% better than those obtained by Simulated Annealing. The ability of any scheme to go to a "good" solution quickly is very important. Figure (3a) compares the convergence of Tabu Search with Simulated Annealing and the interchange method. As we can see from these graphs the Iterative Improvement method depends on the initial partition used, and accordingly the probability to converge to some good local optimum increases as we increase the number of runs. The curve illustrates effectively the overall improvement in cost value with time. Figure (3b) shows the effect of the Aspiration on the overall solution (cutset) using the Chip1 circuit.

Authorized licensed use limited to: University of Johannesburg. Downloaded on March 05,2010 at 03:02:43 EST from IEEE Xplore. Restrictions apply.

Table (4) presents the results obtained using the Simulated Annealing algorithm and a modified Simulated Annealing with memory. The results in Table (4) indicate that a Simulated Annealing version with memory would cut the computation time drastically with the same solution quality achieved by the pure Simulated Annealing algorithm. These results show the effectiveness of Tabu Search as a control mechanism to direct and manage the Simulated Annealing algorithm.

5 CONCLUSION

This paper involved exploring the effectiveness of Tabu Search as a new search methodology to the problem of circuit partitioning in circuit layout VLSI design. The results reveal that the Tabu Search method gives a good compromise between the quality of final partitions and time required to provide the solution. The Tabu Search method yields the best final partition with respect to interchange methods and Simulated Annealing. Initial results on large problems indicate that Tabu Search yields partitions that have at least 10% fewer cut nets than either of these methods.

The paper also explored the effectiveness of combining fundamentally different approaches such as the Tabu Search mechanism and Simulated Annealing. It is evident from the results that the hybridisation approach can avoid many of the weaknesses inherent in the different methodologies, while capitalising on their strengths. Possibilities for merging with other procedures (i.e., Genetic Algorithms and Neural Networks) likewise offer intriguing avenues for exploration for other VLSI circuit layout problems. This is currently under investigation.

REFERENCES

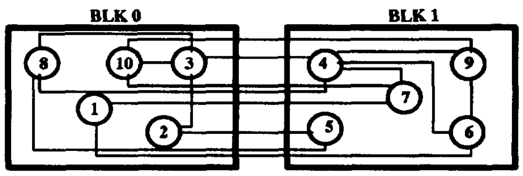
[1] F. Glover. Tabu Search Part I. *ORSA Journal on Computing*, 1(3):190–206, 1990.

[2] M.R Garey and D.S Johnson. *Computers and Intractability*. Freeman, San Francisco CA, 1979.

[3] L.A Sanchis. Multiple-Way Network Partitioning. *IEEE Transactions on Computers*, 38(1):62–81, January 1989.

[4] S. Hadley and B. Mark and A. Vannelli. An Efficient Eigenvector and Node Interchange Approach For Finding Netlist Partitions. *IEEE Transactions on CAD / ICAS*, 11(7):885–892, July 1992.

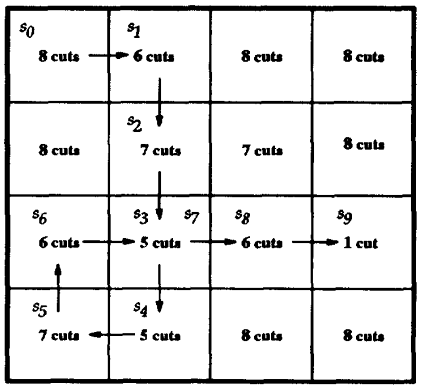
[5] S. Kirkpatrick and C.D. Gelatt and M.P. Vecchi. Optimisation by Simulated Annealing. *Science*, 220(4598):671–680, May 1983.



(a) Initial partition for circuit X.

Iteration	Best Gain	Modudle involved
(1)	+2	Module 5
(2)	-1	Module 4
(3)	+2	Module 3
....		

(b) Gains associated with modules in the circuit



(c) The solution space in terms of cuts

4 MAX VALS	1	2	3	4
MODULE	5	4	3	2
FROM BLK	1	1	0	0
TO BLK	0	0	1	1

(d) Tabu List after 4 iterations

4 MAX VALS	1	2	3	4
MODULE	6	4	3	2
FROM BLK	1	1	0	0
TO BLK	0	0	1	1

(e) Tabu List after the fifth iteration

Aspiration Values Associated with Different Cuts in the Circuit								
Iteration	Cuts							
	1	2	3	4	5	6	7	8
0	hi	hi	hi	hi	hi	hi	hi	hi
1	hi	hi	hi	hi	hi	hi	hi	6
2	hi	hi	hi	hi	hi	7	hi	6
3	hi	hi	hi	hi	hi	7	5	6
4	hi	hi	hi	hi	5	7	5	6
5	hi	hi	hi	hi	5	7	5	6
6	hi	hi	hi	hi	5	7	5	6
7	hi	hi	hi	hi	5	5	5	6

(f) The Aspiration Table

Figure 1: Tabu Moves and Aspiration

Authorized licensed use limited to: University of Johannesburg. Downloaded on March 05,2010 at 03:02:43 EST from IEEE Xplore. Restrictions apply.

```
Input:
    The net list or the Graph G= (V,E)
    K = number of partitions;
    $|T|$ = size of Tabu list
    max_num_iter = maximum iterations
Initialization:
    Initial Partition = Generate a random solution
    $s = (V_1, V_2.., V_k)$
    num_iter = 0; bestpart = s; bestcut = f(s);
Main Loop:
    While num_iter < max_num_iter
        Pick module $m_i$ at random and compute
        $\Delta C$ for the interchange with $m_j$
        Pick best module associated with best gain
        If (move not in TabuList) then
            accept the move
            Update TabuList;
            Update the Aspiration Level;
        End If
        If (move in TabuList) then
            If (Cost(s) < Aspiration($s_0$) then
                Override TabuList Status (accept move)
                Update TabuList and Aspiration;
    End While
OutPut:
    Best Partition, Best Cut
```

Figure 2: Tabu Search Algorithm

Table (1)
Netlist Partitioning Test Cases

Circuit	Nets	Nodes	Node Degree		Net Size	
			$\bar{x}$	σ	$\bar{x}$	σ
Pcb1	32	24	1.1	0.85	1.23	0.95
Chip1	294	300	2.82	1.15	2.87	1.39
Primary1	904	833	3.50	1.29	3.22	2.59
Primary2	3029	3014	3.72	1.55	3.70	3.82
Bio	5711	6417	3.26	1.03	3.66	20.92

Table (2)
2 Way Partitioning

Circuit	INTER		SA		TS	
	Cuts	Time	Cuts	Time	Cuts	Time
Pcb1	5	1.3	5	41.7	5	2.5
Chip1	20	18.7	19	535	18	23.4
Prim1	53	73.3	46	730	42	151.2
Prim2	179	355.9	160	1250	143	400

Table (3)
4 Way Partitioning

Circuit	INTER		SA		TS	
	Cuts	Time	Cuts	Time	Cuts	Time
Pcb1	8	1.83	8	43.5	7	2.94
Chip1	53	37.66	45	1321.8	43	52.2
Prim1	140	137.5	113	4624.8	107	201.3
Prim2	625	887.4	401	16615	392	1032

Table (4)
6 Way Partitioning

Circuit	INTER		SA		TS	
	Cuts	Time	Cuts	Time	Cuts	Time
Pcb1	17	1.6	17	60.24	16	2.95
Chip1	69	52.2	57	1941.9	55	59.6
Prim1	187	199.6	168	6746.5	121	251.2
Prim2	806	2026.8	570	25106	534	2656

Table (5)
Comparison Between SA vs SAM

Blocks	Chip1 Circuit				
	SA		SAM		Speed
	Cuts	Time	Cuts	Time	
2 Blocks	19	535	19	480	10%
4 Blocks	45	1321.8	45	874	73%
6 Blocks	57	1941.9	61	998	48%

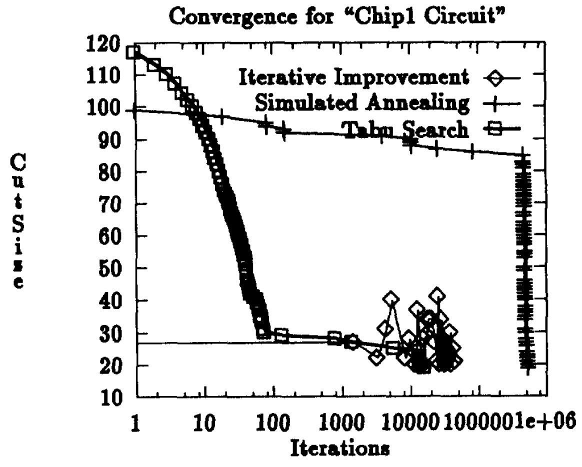


Figure (3a) Convergence

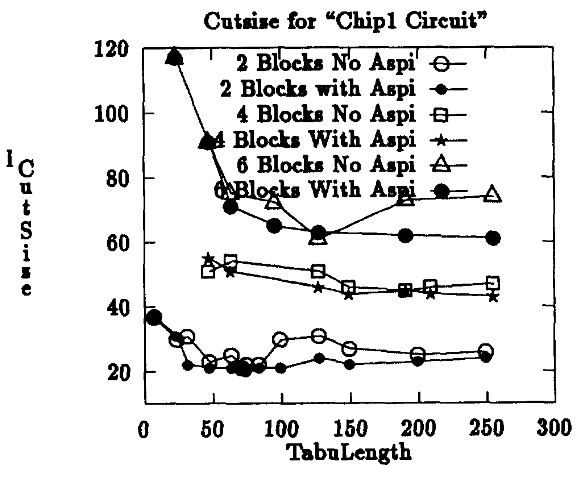


Figure (3b) Aspiration Effect

Authorized licensed use limited to: University of Johannesburg. Downloaded on March 05,2010 at 03:02:43 EST from IEEE Xplore. Restrictions apply.