Application of Semidefinite Programming to Circuit Partitioning*

Changhui Cris Choi
Department of Management Sciences
The University of Iowa
Iowa City, Iowa 52242, U.S.A.

Yinyu Ye
Department of Management Sciences
The University of Iowa
Iowa City, Iowa 52242, U.S.A.

August 23, 1999

Abstract

In this paper we apply Semidefinite Programming (SDP) to solving the circuit partition problem. Unlike other local search methods, we first translate the hypergraph into a weighted (undirected) graph, and then solve it as a graph partition problem using the recent semidefinite program relaxation and heuristic or randomized approximation. Our preliminary computational results indicate that as the number of modules in the circuit increases, the quality of the partition resulted from SDP becomes superior to that resulted from previous methods.

Key words. semidefinite programming, circuit partition, graph partition, semidefinite program relaxation.

*This work was partially supported by NSF grants DMI-9522507 and DMS-9703490.

1 Introduction

VLSI circuit partition techniques are used to reduce the VLSI chip size by carefully replacing components and interconnections in different parts, while enable efficient parallel simulation and communication and minimize time delays among these circuit parts. The circuit bisection is a basic technique in partitioning process. It divides a circuit into two parts with similar sizes, but at the same time minimizes the number of "connections" between the two circuit parts.

Typically, a circuit, C, is represented by a hypergraph or netlist $H(V, E)$ with V representing a set of modules (pins and cells) and E a set of nets. Figure 1 depicts a simple circuit example.

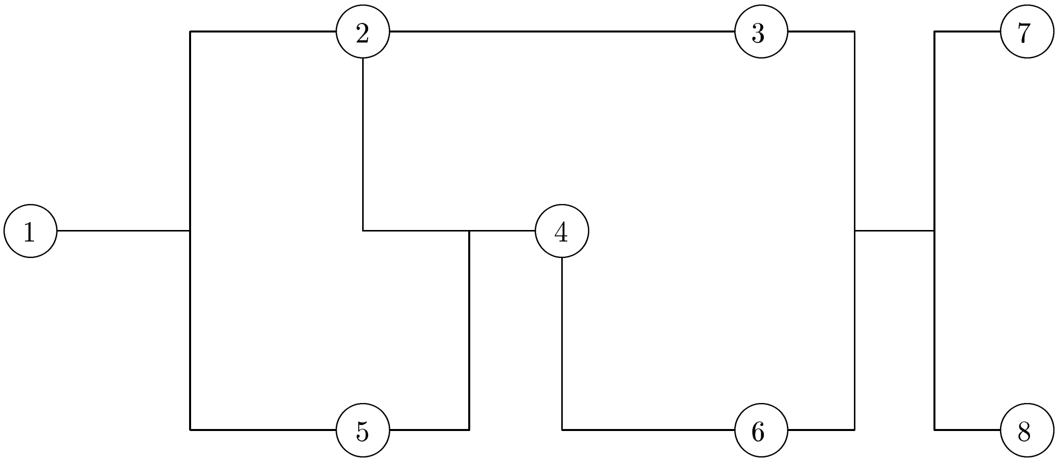

Figure 1 : A simple circuit example.

As shown in Figure 1, some multiple modules of the set V of eight modules in the circuit are connected by single nets, respectively. For example, modules 1, 2 and 5 are connected by a single net, denote by $(1, 2, 5)$. Thus, if modules 1, 2 and 5 were separated in a circuit partition, then this single net would have been "cut". The objective of the partition is to minimize the number of nets or connections being cut. We see here, $(1, 2, 5)$, $(2, 4, 5)$, $(2, 3)$, $(4, 6)$ and $(3, 6, 7, 8)$ constitute set E of all single nets in the circuit.

Each module has certain size or area, denoted by a vector a where a_i denotes the size of module i. In the circuit bisection problem, we need to divide the circuit such that the total size of one of the two parts is about equal to each other.

In this paper we apply Semidefinite Programming (SDP) to solving the circuit bisection problem. Unlike other previous local search methods, we first translate the hypergraph into a weighted and undirected regular graph, and then solve it as a graph partition problem using the recent semidefinite program relaxation and heuristic or randomized approximation. Our preliminary computational results indicate that as the number of modules in the circuit increases, the quality of the partition resulted from SDP becomes superior to that resulted from previous methods.

2 Translating hypergraph

There are many ways to translating hypergraph into weighted and undirected graph. The method we used is to translate a single net into a complete subgraph and each arc in the subgraph assigned the same weight equal to

$$\frac{4}{(\text{number-of-modules-in-net})^2}.$$

For example, the net $(1, 2, 5)$ would become a complete subgraph of node 1, 2, and 5, and each of its three arcs has a weight $4/9$. These complete subgraphs are finally merged into a weighted regular graph, where each arc weight is the sum of weights of all the subgraphs who contain the arc.

Figure 2 depicts the weighted graph translated from the circuit shown in Figure 1.

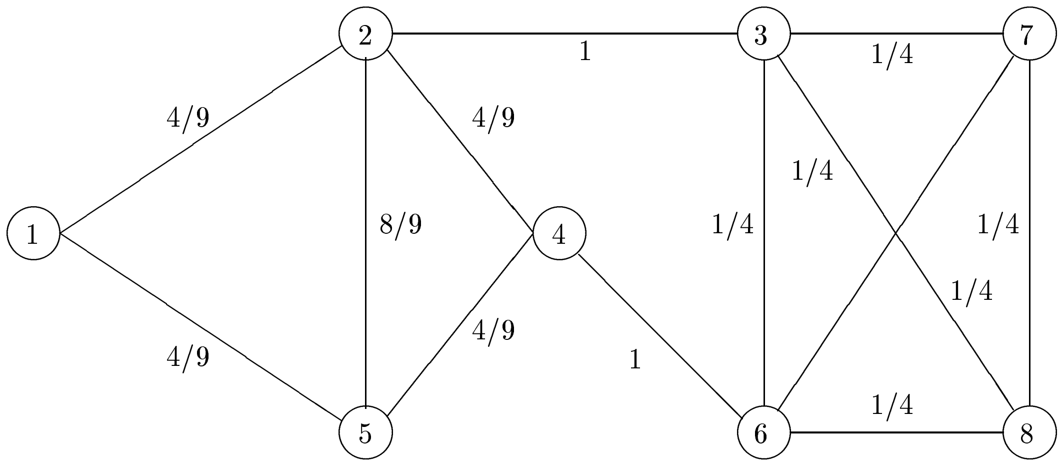

Figure 2 : The weighted graph after translation.

Note that the weight on the arc between node 2 and 5 is the sum of the weights $4/9$ from the net $(1, 2, 5)$ and $4/9$ from the net $(2, 4, 5)$. Again, nodes after the translation are associated with the same size vector a.

We now solve the graph bisection problem for the translated graph, that is, we bisect the nodes into two sets such that each set has about equal total size, and the sum of weights of cut arcs is minimized.

There are many other weighting schemes. The weight assignment we adopted has advantage over the other schemes in the fact that it tends give more freedom to cut bigger nets in the bisection. Since weight in a bigger net is relatively smaller than that of a smaller net, the graph bisection would prefer to cut arcs in the bigger net than in a smaller net to make total sizes on both sides equal. Note that in the original circuit bisection problem, regardless the number of modules in a net, it is counted as one cut net if any module of the net is separated.

3 SDP method

The graph (with n nodes) bisection problem can be solved as a difficult (NP-hard) binary integer programming problem:

$$
\text{(GB)} \qquad \begin{array}{lrl} \text{Min} & x^T L x & \\ \text{Subject to} & a^T x & = 0, \\ & x_i & \in \{-1, 1\}, \quad i = 1, ..., n. \end{array} \tag{1}
$$

Here, a is a vector of node sizes as we described early (for the example we assume that all modules have a same size, that is, a is e, the vector of all ones), L is the Laplacian matrix

$$
L = \text{Diag}(We) - W,
$$

and W is the weighted adjacency matrix for the weighted graph, i.e., w_{ij} is the weight between node i to j and it is 0 when the two nodes are not connected. For the translated graph in Figure 2, matrix W will look like:

$$
W = \begin{pmatrix} 0 & 4/9 & 0 & 0 & 4/9 & 0 & 0 & 0 \\ 4/9 & 0 & 1 & 4/9 & 8/9 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1/4 & 1/4 & 1/4 \\ 0 & 4/9 & 0 & 0 & 4/9 & 1 & 0 & 0 \\ 4/9 & 8/9 & 0 & 4/9 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/4 & 1 & 0 & 0 & 1/4 & 1/4 \\ 0 & 0 & 1/4 & 0 & 0 & 1/4 & 0 & 1/4 \\ 0 & 0 & 1/4 & 0 & 0 & 1/4 & 1/4 & 0 \end{pmatrix}. \tag{2}
$$

However, we solve the following semidefinite program, which is a relaxation of the above integer programming problem and "polynomially" solvable.

$$
\text{(SDP)} \qquad \begin{array}{lrl} \text{Min} & L \bullet X & \\ \text{Subject to} & X_{ii} & = 1, \quad i = 1, ..., n, \\ & (aa^T) \bullet X & = 0, \\ & X & \succeq 0. \end{array} \tag{3}
$$

Here $\bullet$ is the matrix inner product operation, which is the trace of the product of the two symmetric matrices, and $X \succeq 0$ means that X is positive semidefinite. One can view X as xx^T, the out-product of x in problem (GB).

We solve the SDP problem using code DSDP, a dual-scaling semidefinite programming solver written by COPL of the University of Iowa [6, 7]. The dual of the SDP problem is:

$$
(\text{DSDP}) \qquad \begin{array}{ll} \text{Maximize} & \displaystyle\sum_{i=1}^{n} y_i \\ \text{Subject to} & \text{Diag}(y) + \lambda a a^T + S = L, \quad S \succeq 0, \end{array} \tag{4}
$$

where y is a vector with the dimension n (number of nodes in the problem).

4 Rank reduction

After we obtain an approximate optimal X of (SDP), n integral vectors are generated by several rank reduction techniques, similar to [5] and [7], from X. Each component of a vector is either -1 and 1, which represents a bisection of the modules for the circuit. When search for the best bisection among these vectors, we use the original hypergraph file to count the number of cut nets. This is necessary because the SDP relaxation bisects the translated weight graph which is just an approximation of the original hypergraph.

For the example depicted in Figure 1, the SDP produces a vector solution:

$$(-1, -1, 1, -1, -1, 1, 1, 1),$$

that is, set nodes $\{1, 2, 4, 5\}$ on one side and nodes $\{3, 6, 7, 8\}$ on the other, bisecting the circuit as below. This bisection allows only 2 cut nets, which is optimal. Note that for this example we have assumed that all 8 modules have a same size.

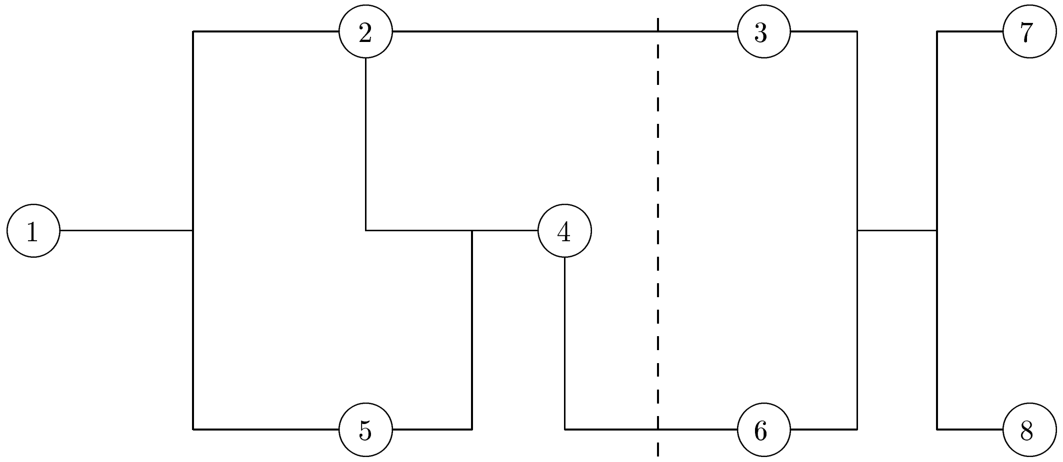

Figure 3 : The bisection of the circuit.

5 Test result

The following Table compares the SDP method introduced in this paper with four other local search and random methods. These test circuit problems (named in column 1) and results are obtained from [4] written by Shantanu Dutt and Wenyong Deng in 1996. In this comparison test, each of theses circuits is partitioned into two equally sized parts. The second column represents the number of modules in each circuit, and columns 3-7 show the number of cut nets by the bisections obtained from these five methods, respectively.

There methods are

1. FM : 100 runs of the “Fidducia-Matthetses” code;
2. LA-3 : the “look ahead” method;
3. Window : one of the clustering based partioner;
4. Prop : the probability based approach proposed by [4];
5. SDP : the semi-definite programming described in this paper.

A quick explanation. The FM code looks for the immediate improvement using local net list information. The “look ahead” method is proposed to improve the FM method, and it literally looks ahead of several steps to seek better solution rather than immediate gain by switching modules.

Problem	number of modules	FM	LA-3	Window	Prop	SDP	sdp_time
p1	833	57	55	60	59	57	0.04
bm1	882	55	55	70	54	55	1.57
t4	1515	86	88	61	58	61	0.2
t3	1607	72	90	67	58	58	0.24
t2	1663	115	105	105	91	107	0.28
t6	1752	71	63	70	81	70	0.46
struct	1952	45	45	n/a	38	37	0.5
t5	2595	97	96	101	82	92	1.41
19ks	2844	142	153	136	120	131	1.57
p2	3014	236	183	258	154	176	2.98
s9234	5866	53	58	n/a	55	50	8.95
biomed	6514	83	91	163	88	86	32.11
s13207	8772	92	89	n/a	83	78	59.1
s15850	10470	112	75	n/a	73	67	88.08
industry2	12637	428	378	392	254	242	423.68

Table 1: Comparison of the SDP with other methods (sdp_time is in hours).

The Window approach is a clustering-based technique used by Charles J. Alpert [3]. The Prop method is a probability-based method proposed by Shantanu Dutt and Wenyong Deng on [4]. Finally, the SDP method is the global semidefinite programming approximation method based on the dsdp code of [6, 7].

One clear trend seen from the table is that the SDP method works better for larger problems. However, its computation time is substantially longer than any of the other methods, which is the subject for further research and improvement.

6 Final remarks

We have proposed a new method based on SDP for circuit partition. Unlike other search methods, our method would search for a global approximate solution in a weighted graph representation. It appears that the SDP method works better than other methods, especially when the circuit is large.

At this point we have not implemented the bisection to allow some difference, say 5%, between the two parts. It is a matter of time that we implement this allowance into the SDP code. This would allow us compare more favorably with other existing methods.

But there are limitations, mainly memory requirement and and solution time. For a circuit with n modules we need to store and solve an $n \times n$ double-precision and dense symmetric matrix, which is the bottle neck in our SDP method. Now we are working on an iterative method that does not generate and solve this matrix explicitly and it looks quite promising so far.

References

[1] C. Berge, *Graphs and Hypergraphs* (American Elsevier, New york, 1976).

[2] Charles J. Alpert and Andrew Kahng, "Recent directions in netlist partitioning: a survey," Ph.D. Thesis, UCLA computer science department, Los Angeles, CA, 1995.

[3] Charles J. Alpert and Andrew Kahng, "A general frame work for vertex orderings, with applications to circuit clustering," Proc. IEEC/ACM international Conference on CAD, November 1994, pp 63-67.

[4] Shantanu Dutt and Wenyong Deng, "A probability-based approach to VLSI circuit Partitioning," Design automation conference report, Department of electrical engineering, The University of Minnesota, Minneapolis, MN, 1996.

[5] Steve John Benson, "Solving large scale combinatorial optimization problems," Ph.D Thesis, Department of Mathematics, The University of Iowa, Iowa City, 1998.

[6] S. Benson, Y. Ye, and X. Zhang, "Solving sparse, large scale positive semidefinite programs," Department of Management Science, The University of Iowa, Iowa City, 1997, To appear in *SIAM J. of Optimization.*

[7] S. Benson, Y. Ye, and X. Zhang, "Mixed Linear and Semidefinite Programming for Combinatorial and Quadratic Optimization," Department of Management Science, The University of Iowa, Iowa City, February 13, 1998; revised in April 1999.

[8] Y. Ye, *Interior Point Algorithms : Theory and Analysis* (Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons, New York, 1997).