CLUSTERING TO IMPROVE BI-PARTITION QUALITY AND RUN TIME

Gregory Tumbush

Air Force Research Labs
AFRL/IFTA
2241 Avionics Circle STE 32
WPAFB, OH 45433-7334
tumbusgj@sensors.wpafb.af.mil

Dinesh Bhatia

Design Automation Laboratory
University of Cincinnati
Cincinnati, OH 45221-0030
Dinesh.Bhatia@uc.edu

ABSTRACT

Circuit partitioning is a very extensively studied problem. Traditional heuristic methods used to solve these problems degrade in performance when the problem size becomes large. Clustering has been proposed as a method to reduce the problem size, allowing the problem to be solved faster and with more accuracy. In [7] we propose a novel bi-partitioning method based on a mathematical formulation of the problem which can be solved using commercial combinatorial optimization (CO) tools. In this paper we continue to formulate the problem as a nonlinear program (NLP) but also apply clustering to determine if run-time and quality can be improved.

1. INTRODUCTION

Given a set of n netlist modules $V = \{v_0, v_1, \ldots, v_{n-1}\}$ and m pairs of distinct vertices called *edges* $E=\{e_1,e_2,\ldots, e_m\}$ we may represent the circuit as a graph $G = (V, E)$. A vertex $v_x, x = 0, \ldots, n-1$ is adjacent to a vertex $v_y, y = 0, \ldots, n-1$ if (v_x, v_y) is an edge, i.e., $(v_x, v_y) \in E$. The goal of bi-partitioning is to assign each $v_i, i = 0, \ldots, n-1$, to one of two segments such that the *capacity* of neither segment is violated and the *cutset* is minimized. An edge e is *cut* if all the terminals of e are not within a single segment. The total number of cut edges is called the size of *cutset*.

Clustering will produce a set of k disjoint, nonempty clusters $C_1, C_2, \ldots, C_k$ such that $C_1 \cup C_2 \cup \ldots \cup C_k = V$. This reduces the problem instance to size k instead of n. The clustering process will also decrease the sparsity of the netlist. It is believed that the FM and KL partitioning algorithms will perform better on netlists of higher density and smaller solution space[2][4]. In this way, the ordering of modules (nodes) based on clustering creates a natural bias towards the search for a better solution. As problem instances grow, clustering will be an integral component of any partitioning algorithm, particularly move based approaches. We use a clustering algorithm called CL[3] that uses the *geometric embedding* of a graph G to find a good clustering.

One method to integrate clustering into a partitioning algorithm is via the so called *two-phase method*. In this method, the partitioning algorithm is executed on the contracted netlist(phase 1) and the solution is the starting point for the second execution on the original (flat) netlist(phase 2). This is the method we will use to integrate CL into our partitioning algorithm.

In this paper we will examine the effects of clustering to improve on the quality of a bi-partition. Using the FM method we will perform bi-partitioning on *flat* gate level benchmarks. We then perform two-phase bi-partitioning using the FM method and a new bi-partitioning model and compare the results.

1.1. Combinatorial Optimization

Generally a combinatorial optimization (CO) problem is an optimization problem of the form

$$\begin{aligned} &minimize && f(x) \\ &subject\ to && g_i(x) \leq 0 \quad i = 1, \ldots, m. \end{aligned} \tag{1}$$

The function $f(x)$ is the objective function and the set of conditions $g_i(x) \leq 0, i = 1, \ldots, m$, are the constraints of the problem. Note that the number of constraints can be very large. Every vector x that satisfies the constraints is a *solution* to the problem. A solution that minimizes $f(x)$ over the set of all solutions is an *optimal solution*. A vector x' is a *local optimum* if and only if there exists a neighborhood $V(x')$ of x' such that x' is a *global optimum* of the problem.

The forms that $f(x)$ and $g_i(x)$ take determine the type of CO problem. If $f(x)$ is *linear*, the problem is a linear program(LP), while if $f(x)$ is *non-linear*, the problem is a non-linear program(NLP). $f(x)$ may have both linear and non-linear elements. The constraints $g_i(x)$ can also be linear, non-linear, or a combination of both. A special case

0-7803-5471-0/99/$10.00©1999 IEEE

of non-linear CO is when the terms of $f(x)$ are quadratic and the constraints are linear. This is called a *quadratic program*(QP). If $f(x)$ does not exist and only constraints are present the problem becomes a *constraint satisfiability problem* (CSP). Equations 2 and 3 show the forms of linear and quadratic programs respectively.

$$minimize \quad c^T x \quad subject\ to\ Ax \leq b \tag{2}$$

$$minimize \quad \frac{1}{2}x^T Dx + c^T x\ subject\ to\ Ax \leq b \tag{3}$$

In equations 2 and 3, c^T is the transpose of the coefficients of the linear optimization function, x is the solution variables, A is the constraint matrix of the linear constraints, b is the right hand side of the linear constrains, D is the coefficient matrix of the quadratic optimization function, and x^T is the transpose of x.

Many fast methods exist to solve a LP[5] as do methods to solve a NLP if it is *convex*, that is, every local optimum is also a global optimum. However, there are no known methods to find a global optimum for a non-convex NLP problem. Only a local optimum is guaranteed to be found in this case. The graph bi-partitioning problem when formulated as a CO problem is non-convex.

We solve our problem with the commercial LP/NLP solver MINOS 5.4. MINOS can solve large scale linear and non-linear programs and takes advantage of sparsity of matrices[6].

2. CLUSTERING TO IMPROVE BI-PARTITIONING QUALITY

The two-phase bi-partitioning method seeks to increase partition quality on large circuits by clustering the circuit and then bi-partitioning the clustered circuit. The results of the bi-partition are used as a starting point for bi-partitioning the *flat* circuit.

Phase 1 of two-phase bi-partitioning will perform bi-partitioning on the clustered netlist. For each benchmark, the CL clustering algorithm creates 10 distinct clustered netlists, corresponding to 10 embedding dimensions. For each embedding dimension d, $1 \leq d \leq 10$, a bi-partition of the flattened netlist is found by expanding the bi-partitioned clusters. This bi-partition is used in phase 2 as the initial bi-partition.

In phase 2, the flat netlist is bi-partitioned using the initial bi-partition found in phase 1. For each embedding dimension d, $1 \leq d \leq 10$, a bi-partition of the flattened netlist is found, utilizing the corresponding initial bi-partition.

The CO problem is solved as an assignment problem. We associate a variable $x_i, 0 \leq i \leq n-1$ for n components. It is predetermined for bi-partitioning that if $x_i = 1$ then module i belongs to a partitioning segment and to the complimentary one if $x_i = 0$. A solution to the NLP problem can result in non-integer assignment to x_i which will not form a feasible partitioning solution. Thus, fractional assignment variables have to be rounded for generating a feasible partitioning solution. We employ 0-1 rounding for changing the fractional assignments to an integer form. This can be done simply by choosing a value, *median*, and if $x_i \geq median$ set x_i to 1, 0 otherwise.

2.1. Problem and Solution

Consider a three cell net, 0, 1, and 2, to be bi-partitioned and it's associated assignment variables x_0, x_1, and x_2. The graph cutset of this net will be:

$$cutset = (x_0 + x_1 - 2x_0x_1) + (x_0 + x_2 - 2x_0x_2) + \tag{4}$$

$$(x_1 + x_2 - 2x_1x_2). \tag{5}$$

In general the graph cutset is

$$\sum_{\forall r \in M} \sum_{i=1}^{|Q_r-1|} \sum_{j=i+1}^{|Q_r|} x_{Q_{ri}} + x_{Q_{rj}} - 2x_{Q_{ri}}x_{Q_{rj}} \tag{6}$$

where Q_r is the ordered set of all non I/O elements on net r, such that Q_{r1} is the first element, Q_{r2} is the second element,..., $Q_{r|Q_r|}$ is the last element and M is the set of all nets. This will form $f(x)$ in equation 1.

Let $a_i, i = 0, \ldots, n-1$, be the area of cell i. For bi-partitioning, the area constraint on segment 1 and 2 is

$$\sum_{i=0}^{n-1} a_i x_i \leq A_1 \tag{7}$$

$$\sum_{i=0}^{n-1} a_i (1 - x_i) \leq A_2$$

where A_1 and A_2 are the capacity constraints on two partitioning segments, 1 and 2 respectively. This will form $g_1(x)$ and $g_2(x)$ in equation 1. Note that A_1 and A_2 are not necessarily the same.

3. RESULTS AND ANALYSIS

In this section we present results demonstrating the effects of clustering on bi-partition quality. To determine if clustering provides any benefit we also perform bi-partitioning on the flat benchmarks using the FM method. We also show to what extent an initial starting point affects the bi-partitioning model solved by MINOS. The characteristics of these benchmarks[1] appear in Table 1. The clusters induced by the CL algorithm on each benchmark for each of 10 embedding dimensions were provided by C. Alpert[1].

We will use the FM method and the bi-partitioning model solved by MINOS to perform two-phase bi-partitioning. We

[1]The Circuit Partitioning Page http://vlsicad.cs.ucla.edu/ cheese/benchmarks.html

wish to separately test the effect that MINOS and FM exhibit in both phases of two-phase bi-partitioning. Also, we will test the sensitivity of MINOS to the initial partition. This is accomplished by running MINOS using the default initial starting point, i.e. all variables equal to 0. This corresponds to all cells being initially assigned to 1 partition. Only one run of MINOS is required for each embedding dimension in this case. Therefore, there will be 6 different ways to apply FM and MINOS to two-phase bi-partitioning.

For each embedding dimension, 20 runs of FM, 20 runs of MINOS and 1 run of MINOS are used to create the bi-partitioned clusters for a total of 410 runs for each benchmark. The random initial partition created by FM is used as a starting point for each of the 20 runs of MINOS. In phase 2 the 410 initial partitions created in phase 1 are used as an initial starting point for bi-partitioning the flat netlist. Both FM and MINOS are used to bi-partition the flat netlists for a total of 820 runs.

3.1. Analysis

Table 2 compares the best cutsize of 50 runs of FM on an unclustered flat netlist to the best cutsize of two-phase bi-partitioning. Using the FM method or 20 runs of MINOS for phase 1 and then FM for bi-partitioning the flat netlist resulted in the most improvement over flat bi-partitioning. The average decrease in cutset is 8.6%. When only 1 run of MINOS was used for phase 1 and then FM was used to bi-partition the flat netlist the cutsize increased by 15%. We did not observe as large of an improvement as in [1] but we used different module sizes and a different area constraint. This resulted in better partitioning results than in [1] for 50 runs of FM on a flat netlist. The other three combinations of methods for two-phase bi-partitioning produced inferior results.

Table 3 compares the runtimes of 50 runs of FM on an unclustered flat netlist to the runtimes of two-phase bi-partitioning. Using the FM method or 1 run of MINOS for phase 1 and then FM for bi-partitioning the flat netlist resulted in the most improvement over flat bi-partitioning. The average decrease in runtime is 94% and 99% respectively. The other four combinations of methods for two-phase bi-partitioning, while faster than 50 runs of FM, were appreciably slower. Also, note that using 1 run of MINOS instead of the FM method in phase 1 and then the FM method in phase 2 resulted in an average decrease in runtime of 74%.

These results imply that clustering should *always* be utilized. Not only does clustering reduce the runtime but the cutsize is also improved. In all cases the FM method should be used for phase 2. However, if time is critical 1 run of MINOS should be used in phase 1, otherwise the FM method should be used to create the initial partition.

4. SUMMARY

In this paper we present a mathematical model for the graph bi-partitioning problem and demonstrate the effects clustering has on run-times and cutsizes. To evaluate the merits of FM and MINOS in two-phase bi-partitioning we tested all possible combinations and compared the results to flat bi-partitioning using the FM method. This demonstrates the effects of clustering to improve not only the quality of bi-partitioning but also the runtimes.

5. REFERENCES

[1] C. J. Alpert and A. B. Kahng. Multiway partitioning via geometric embeddings, orderings, and dynamic programming. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 14(11):1342–1357, November 1995.

[2] M. K. Goldberg and M. Burstein. Heuristic improvement technique for bisection of VLSI networks. In *Proceedings of the IEEE International Conference on Computer Design*, pages 122–125, 1983.

[3] S. C. Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

[4] T. Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout*. Wiley-Teubner, 1990.

[5] M. Minoux. *Mathematical Programming Theory and Applications*. Wiley-Interscience, 1986.

[6] J. J. More' and S. J. Wright. *Optimization Software Guide*. SIAM, 1993.

[7] G. Tumbush and D. Bhatia. Partitioning under timing and area constraints. In *Proceedings of the IEEE 1997 International Conference on Computer Design*, October 1997.

Table 1: Benchmark Characteristics

Bench Mark	Number Cells	Number Nets	Total Area	Largest Cell
t2	1663	1720	261,285,092	64,025,484
t3	1607	1618	136,599,614	18,057,076
t4	1515	1658	172,532,995	67,954,944
t5	2595	2750	305,422,515	114,264,400
t6	1752	1641	123,505,052	1,351,394
p1	833	902	27,982,000	180,000
p2	3014	3029	52,324,000	180,000

Table 2: Decrease in Cutset by Use of Clustering

Bench Mark	Phase 1 Solved by FM	Phase 1 Solved by MINOS-20 Runs	Phase 1 Solved by MINOS-1 Run	Phase 1 Solved by FM	Phase 1 Solved by MINOS-20 Runs	Phase 1 Solved by MINOS-1 Run
	Phase 2 solved by FM			Phase 2 solved by MINOS		
t2	0%	0%	0%	-40%	-40%	-81%
t3	13%	13%	-3%	-1%	24%	-49%
t4	13%	4%	-75%	-63%	-171%	-171%
t5	5%	5%	5%	-90%	-90%	-90%
t6	13%	17%	-11%	-30%	-13%	-20%
p1	0%	2%	-31%	-31%	-20%	-63%
p2	16%	18%	12%	-9%	11%	13%

Table 3: Decrease in Runtime by Use of Clustering

Bench Mark	Phase 1 Solved by FM	Phase 1 Solved by MINOS-20 Runs	Phase 1 Solved by MINOS-1 Run	Phase 1 Solved by FM	Phase 1 Solved by MINOS-20 Runs	Phase 1 Solved by MINOS-1 Run
	Phase 2 solved by FM			Phase 2 solved by MINOS		
t2	83%	-99%	98%	69%	-112%	89%
t3	96%	81%	99%	68%	54%	76%
t4	95%	73%	99%	77%	62 %	90%
t5	98%	46%	99%	72%	34%	73%
t6	96%	-3%	98%	57%	10%	96%
p1	96%	71%	99%	84%	62%	88%
p2	93%	-47%	98%	-60%	-114%	49%