VLSI Design
1995, Vol. 3, No. 1, pp. 93–98
Reprints available directly from the publisher
Photocopying permitted by license only

New Methods for the Construction of Test Cases for Partitioning Heuristics

YOUSSEF SAAB
Computer Science Department, University of Missouri-Columbia, Mathematical Sciences Building, Columbia, MO 65211

(Received April 9, 1993, Revised July 30, 1993)

Partitioning is an important problem in the design automation of integrated circuits. This problem in many of its formulation is NP-Hard, and several heuristic methods have been proposed for its solution. To evaluate the effectiveness of the various partitioning heuristics, it is desirable to have test cases with known optimal solutions that are as "random looking" as possible. In this paper, we describe several methods for the construction of such test cases. All our methods except one use the theory of network flow. The remaining method uses a relationship between a partitioning problem and the geometric clustering problem. The latter problem can be solved in polynomial time in any fixed dimension.

Key Words: *Bisection; Geometric clustering; Graphs; Heuristics; Partitioning*

1. INTRODUCTION

Partitioning has proven to be a fundamental problem in the design automation of integrated circuits, and especially in the layout area [1–6]. Partitioning in several of its formulations is NP-Hard [7]. Therefore, the various existing partitioning methods are heuristics in nature [8–15]. In analyzing the effectiveness of partitioning methods, it is useful to have test cases (inputs) for which an optimal solution is known. In this paper, we restrict our attention to one important formulation of the partitioning problem, which we will refer to as the minimum bisection problem (MBP). Informally, MBP seeks to partition a graph into two parts of about equal sizes such that the sum of weights on the edges cut by the partition is minimized. We will describe 4 methods for the generation of test cases with known optimal solutions for MBP. Three of these methods use the network flow theory [16], and the fourth method uses a relationship between MBP and the geometric clustering problem (GC) [13, 17]. The latter problem can be solved in polynomial time in any fixed dimension [18]. In all our methods we restrict ourselves to partitions of equal sizes.

The remainder of this paper consists of five more sections. Section 2 describes some preliminary notations and definitions. Section 3 reviews the Krishnamurthy-Mellema method for the generation of test cases for MBP. Section 4 describes 3 methods for the generation of test cases for MBP using network flow theory. Section 5 describes a method for the generation of test for MBP using a polynomial time algorithm for GC. Finally, Section 6 concludes this paper.

2. PRELIMINARIES

We assume that the reader is familiar with the notion of a set, multiset, and a graph. The number of elements of a set or a multiset A is denoted by $|A|$. The *sum* of two multisets A and B is the multiset $A + B$ consisting of all elements of A and all elements of B. Thus, $|A + B| = |A| + |B|$. Repeated elements of a multiset A can be considered distinct by giving them distinct labels. In this case, the multiset can be considered a set. A mapping f: $A \rightarrow \mathbf{R}$ from a multiset A (considered as a set) to the reals is a *weighting function* on A. Given f, the weight of an element $a \in A$ is $f(a)$, and the weight of a subset $B \subseteq A$ is $\sum_{b \in B} f(b)$. By a slight abuse of notations, we write $f(B)$ for the weight of $B \subseteq A$. A graph $G(V, E)$ is *vertex-weighted* if there exists a

weighting function S defined on V. For $v \in V$, we say that $S(v)$ is the size of v. Similarly for $B \subseteq V$, we say that $S(B)$ is the size of B. In this paper, the sizes of the vertices of a vertex-weighted graph are all assumed to be strictly positive. A graph $G(V, E)$ is *edge-weighted* if there exists a weighting function W defined on E. In the graphs considered in this paper, it is allowed to have repeated (parallel) edges among pairs of vertices, i.e., the set of edges E is a multiset. However, for partitioning purposes, parallel edges can be replaced by a single edge having a weight equal to the sum of their weights. Self-loops do not play any role in partitioning problem, and therefore they can be ignored if they do exist in a graph. A *weighted* graph is a graph which is both vertex-weighted and edge-weighted. Note that any graph can be considered a weighted graph by simply assigning unit weights to its vertices and edges. Given a graph $G(V, E)$, we use $S(v)$ for the size of a vertex $v \in V$, and $W(e)$ for the weight of an edge $e \in E$. The *sum* of two graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ is the graph $G(V, E)$ given by $V = V_1 \cup V_2$ and $E = E_1 + E_2$. In this case, we write $G = G_1 + G_2$. Clearly, for $l > 2$ the operation $G = G_1 + G_2 + \cdots + G_l$ is well-defined, since the operations $\cup$ and $+$ are both commutative and associative. A *partition* of a graph $G(V, E)$ is a partition (V_1, V_2) of the vertex set V, i.e., $V_1 \cap V_2 = \emptyset$, $V_1 \cup V_2 = V$, and both V_1 and V_2 are non-empty. If $S(V_1) = S(V_2)$, then we say that the partition (V_1, V_2) is a *bisection* of G. An edge u—v is said to be *cut* by a partition (A, B) of G if $u \in A$ and $v \in B$. The *cost* of a partition (A, B) of G is the sum of the weights of the edges cut by (A, B) and is denoted by *cost* (A, B). The minimum cut problem (MCP) seeks a minimum-cost partition of a graph, and the minimum bisection problem (MBP) seeks a minimum-cost bisection of a graph. Using network flow techniques [16], MCP can be solved efficiently in polynomial time. However, MBP is NP-hard [7].

3. THE KRISHNAMURTHY-MELLEMA METHOD

This method appeared in [19, 20] for the case of a hypergraph. Here, we consider its application to the case of a graph. The description of this method here is slightly different than in the original papers [19, 20]. However, the basic idea is still the same. We will first establish a general lemma which captures the underlying principle of the Krishnamurthy-Mellema method.

Lemma 1: Let $G(V, E) = G_1(V_1, E_1) + G_2(V_2, E_2) + \cdots + G_l(V_l, E_l)$ be the sum of $l \geq 2$ weighted graphs such that the sets V_i are weighted by the same function $S: V \to \mathbf{R}$. Let c be the minimum cost of any bisection of G, and let c_i, $1 \leq i \leq l$ be the minimum cost of any partition of G_i. If every bisection (A, B) of G is such that $(A \cap V_i, B \cap V_i)$ is a partition of G_i for $1 \leq i \leq l$. Then,

$$c \geq \sum_{i=1}^{l} c_i. \qquad (*)$$

Proof: Consider a bisection (A, B) of G of minimum cost c. Since $G = G_1 + \cdots + G_l$, $cost(A, B) = \sum_{i=1}^{l} cost(A \cap V_i, B \cap V_i)$. But for each $1 \leq i \leq l$, we have $cost(A \cap V_i, B \cap V_i) \geq c_i$. Therefore, $c \geq \sum_{i=1}^{l} c_i$. □

The Krishnamurthy-Mellema method implicitly uses Lemma 1. In this method a graph $G(V, E)$ is constructed as a sum of a number of other graphs $G_i(V_i, E_i)$, $1 \leq i \leq l$, and such that G satisfies the hypothesis of Lemma 1 and admits a bisection (A, B) such that $(A \cap V_i, B \cap V_i)$ is a minimum-cost partition of G_i, $1 \leq i \leq l$. Thus, equality is achieved in (*) and $cost(A, B)$ must be the minimum cost of any bisection of G. In the original Krishnamurthy-Mellema method only unit weights are used for the edges. However, this method can be extended to construct graphs of non-unit weights in a trivial way. Here is an algorithmic description of the Krishnamurthy-Mellema method:

Step 1. Let n be the desired number of vertices, and let k be the desired optimal cost.

Step 2. Let V be a set of n distinct vertices. Consider any partition (A, B) of V. Assign weights to the vertices such $S(A) = S(B) = S(V)/2$. Set counter $i = 1$.

Step 3. Randomly choose a subset V_i of V such that $A_i = A \cap V_i \neq \emptyset$, $B_i = B \cap V_i \neq \emptyset$, and $S(V_i) > S(V)/2$. Set the edge set $E_i = \emptyset$.

Step 4. Randomly choose a vertex $a \in A_i$ and $b \in B_i$ and set $E_i = E_i \cup \{a$—$b\}$.

Step 5. Add enough edges to E_i so that $G_i(V_i, E_i)$ becomes connected. The edges added in this step are **not** cut by the partition (A_i, B_i).

Step 6. If $i < k$ then increment i and go to step 3.

Step 7. Set $G(V, E) = G_1(V_1, E_1) + G_2(V_2, E_2) + \cdots + G_k(V_k, E_k)$. In this step, we assume that $V = \bigcup_{i=1}^{k} V_i$. If not, the remaining vertices in V are included in G as isolated vertices.

Clearly, any bisection (X, Y) of G is such that $(X_i, Y_i) = (X \cap V_i, Y \cap V_i)$ is a partition of G_i for $1 \leq i \leq k$, since $S(V_i) > S(V)/2$ for $1 \leq i \leq k$. Also, the minimum cost of a partition of G_i is 1 for $1 \leq i \leq k$. Therefore, by Lemma 1, the minimum cost of any bisection of G is at least $\sum_{i=1}^{k} 1 = k$. But the bisection (A, B) chosen by the Krishnamurthy-Mellema method has a cost equal to k, since only the k edges chosen in step 4 are cut by (A, B). Consequently, k must be the minimum cost of any bisection of G.

The above method is straightforward. However, it has two disadvantages. First, the number of vertices in the set V_i chosen in step 3 is $\Omega(n)$ under moderate assumptions. Hence, $\Omega(n)$ edges are added in step 5 to guarantee the connectedness of G_i. Consequently, the number of edges in the resulting graph G is $\Omega(nk)$ and is therefore proportional to k. This is not a desirable fact, since in a random-looking graph no clear relationship should exist between the optimal bisection cost and the number of edges. The number of edges in G can be reduced by replacing every p parallel edges by a single one of them having weight p. However, even in this case, the graph is still less random, since we know that with a very high probability G admits a bisection which cuts only unit weight edges. Thus, if $e = u$—v has weight greater than 1, then u and v can be forced to be in the same part while keeping a high probability of finding an optimal bisection. Thus, a special algorithm can be developed to find an optimal bisection in G. The second disadvantage of the Krishnamurthy-Mellema method is that it does not handle edges of different weights. The obvious extension of this method to handle different weights is to give all the edges chosen in steps 4 and 5 the same weight α_i, $1 \leq i \leq k$. Then, the cost of an optimal bisection of G becomes $\sum_{i=1}^{k} \alpha_i$ rather than k. But again G has a non-random structure since the edge set of G can be partitioned into k disjoint parts $E_1, E_2, \ldots, E_k$ such that the weight of any edge in E_i is α_i for $1 \leq i \leq k$.

4. METHODS BASED ON NETWORK FLOW THEORY

The methods in this section rely on the fact that MCP can be solved in polynomial time using network flow techniques [16]. The input to all the methods in this section consists of an integer n. The number of vertices of the generated graphs will be $O(n)$.

Method NF1:

Step 1. Let $G(V, E)$ be a random edge-weighted graph on n vertices.

Step 2. Find a partition (A, B) of G of minimum cost c.

Step 3. Assign weights to the vertices of G such that $S(A) = S(B) = S(V)/2$.

Lemma 2: The minimum cost of any bisection of the graph G generated by Method NF1 is c.

Proof: It suffices to show that (A, B) is a bisection of minimum cost. By step 3, (A, B) is a bisection of G. By step 2, $cost(A, B) = c$ and c is the minimum cost of any partition of G. □

Method NF1 may be disadvantageous, since the cardinality of one of the subsets A or B may be much less than the cardinality of the other subset. Therefore, to enforce $S(A) = S(B)$ in step 3 of Method NF1, much larger weights must be assigned to the vertices of the subset of smaller cardinality. The next two methods avoid the disadvantage of Method NF1.

Method NF2:

Step 1. Let $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ be two random edge-weighted graphs on n vertices each. Without loss of generality, assume that $V_1 \cap V_2 = \varnothing$.

Step 2. Find a partition (A_1, B_1) of G_1 of minimum cost c_1.

Step 3. Find a partition (A_2, B_2) of G_2 of minimum cost of c_2.

Step 4. Set $G(V, E) = G_1(V_1, E_1) + G_2(V_2, E_2)$.

Step 5. Assume without loss of generality that $|A_1| \geq |B_1|$ and $|A_2| \leq |B_2|$. Randomly add enough edges to G between vertices of A_1 and A_2 and enough edges between vertices of B_1 and B_2, until the partition (V_1, V_2) in G has cost $c_1 + c_2$.

Step 6. Let $A = A_1 \cup A_2$ and $B = B_1 \cup B_2$. Assign weights to the vertices of G so that $S(A) = S(B) = S(V_1) = S(V_2) = S(V)/2$. This is always possible (e.g., $S(A_1) = S(B_2) = x$ and $S(A_2) = S(B_1) = y - x$, where $0 < x < y$).

Lemma 3: $c_1 + c_2$ is the minimum cost of any bisection of the graph G generated by Method NF2.

Proof: It suffices to show that (A, B) is a minimum cost bisection of G. By step 6, $S(A) = S(B) = S(V)/2$, and hence (A, B) is a bisection of G. All

the edges added in step 5 are not cut by (A, B). Therefore, by steps 2 and 3, $cost(A, B) = c_1 + c_2$. Consider any other bisection (A', B') of G. If $A' = V_1$ then $B' = V_2$. It follows from step 5 that $cost(A', B') = c_1 + c_2$. Similarly, if $A' = V_2$ then $B' = V_1$, and $cost(A', B') = c_1 + c_2$. Therefore, we can assume that $A' \neq V_1$ and $A' \neq V_2$. Since (A', B') and (V_1, V_2) are both bisections of G, we must have $V_1 \cap A' \neq \emptyset$, $V_1 \cap B' \neq \emptyset$, $V_2 \cap A' \neq \emptyset$, and $V_2 \cap B' \neq \emptyset$. Also, $(V_1 \cap A', V_1 \cap B')$ is a partition of G_1, and $(V_2 \cap A', V_2 \cap B')$ is a partition of G_2. By steps 2 and 3, $cost(V_1 \cap A', V_1 \cap B') \geq c_1$ and $cost(V_2 \cap A', V_2 \cap B') \geq c_2$. But every edge cut by either $(V_1 \cap A', V_1 \cap B')$ in G_1, or $(V_2 \cap A', V_2 \cap B')$ in G_2, is also cut by (A', B') in G. Therefore, $cost(A', B') \geq c_1 + c_2$. □

Let $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ be two graphs such that $|V_2| \geq |V_1|$, and consider the following operation:

Step 1. Find an injective mapping $f: V_1 \rightarrow V_2$.

Step 2. For every edge u—v in G_1, put a corresponding edge $f(u)$—$f(v)$ in G_2 and assign to it the same weight as u—v in G_1.

By performing the above operation, we have actually created a subgraph of G_2 which is isomorphic to G_1. We say that we have embedded G_1 into G_2 by f. The next method uses embedding operations to find a graph with a known cost for the optimal bisection.

Method NF3:

Step 1. Let $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ be two random graphs of n vertices each.

Step 2. Find a partition (A_1, B_1) of G_1 of minimum cost c_1.

Step 3. Find a partition (A_2, B_2) of G_2 of minimum cost c_2.

Step 4. Assume without loss of generality that $|A_1| \geq |B_1|$ and $|A_2| \leq |B_2|$. Consider an empty graph $G(V, \emptyset)$ such that $|V| = |A_1| + |B_2|$. Let (A, B) be a partition of G such that $|A| = |A_1|$ and $|B| = |B_2|$.

Step 5. Embed G_1 in G by an injective mapping f_1: $V_1 \rightarrow V$ such that $f_1(A_1) = A$ and $f_1(B_1) \subseteq B$.

Step 6. Embed G_2 in G by an injective mapping f_2: $V_2 \rightarrow V$ such that $f_2(A_2) \subseteq A$ and $f_2(B_2) = B$.

Step 7. Assign weights to the vertices in V such that $S(A) = S(B) = S(V)/2$.

Lemma 4: $c_1 + c_2$ is the minimum cost of any bisection of the graph G generated by Method NF3.

Proof: By steps 5 and 6, we have $cost(A, B) = c_1 + c_2$. Let $X = f_1(V_1)$ and let $Y = f_2(V_2)$. By steps 5 and 7, $S(X) = S(A) + S(f_1(B_1)) > S(A) = S(V)/2$. Similarly, by steps 6 and 7, $S(Y) = S(B) + S(f_2(A_2)) > S(B) = S(V)/2$. Thus in any bisection (A', B') of G, we must have $X_1' = X \cap A' \neq \emptyset$, $X_2' = X \cap B' \neq \emptyset$, $Y_1' = Y \cap A' \neq \emptyset$, and $Y_2' = Y \cap B' \neq \emptyset$. Also, (X_1', X_2') is a partition of the subgraph of G that is isomorphic to G_1 by f_1, and (Y_1', Y_2') is a partition of the subgraph of G that is isomorphic to G_2 by f_2. Therefore, by steps 2 and 3, $cost(X_1', X_2') \geq c_1$ as an isomorphic partition of G_1 (i.e., here we only sum the weights of the edges of the subgraph that is isomorphic to G_1 in G), and $cost(Y_1', Y_2') \geq c_2$ as an isomorphic partition of G_2. Consequently, since the set of edges of the two subgraphs that are isomorphic to G_1 and G_2 in G, are disjoint by construction, $cost(A', B') \geq c_1 + c_2$. □

5. A METHOD BASED ON GEOMETRIC CLUSTERING

In this section, we describe a method for the generation of test case for MBP using an algorithm for the solution of the geometric clustering problem (GC). The relationship of MBP and GC is due to Frankle and Karp [13]. The method described here relies on an algorithm for GC, which runs in polynomial-time in any fixed dimension. This algorithm is due to Montgomery-Smith and Saab [18], and a similar algorithm was independently discovered by Arun and Rao [21] about the same time. The theorem which is the basis of the algorithm by Montgomery-Smith and Saab allows the generated graphs to have only unit sizes for the vertices, and it does not allow for parallel edges.

It is useful here to review the matrix representation of simple graphs, where a simple graph is one that does not admit parallel edges.

The connection matrix C of a simple graph is defined by $c_{ij} = 0$ if there exist no edge between vertices i and j. Otherwise, c_{ij} is equal to the weight of the edge i—j. For MBP, self-loops do not matter, so they can be ignored if they do exist in the graph.

Given an even number n of points (vectors) in d-dimensional Euclidean space, GC seeks to find a partition of these n points into two sets S_1 and S_2 of equal cardinality such that the Euclidean distance between the centroids of the points in S_1 and S_2 is

maximized, where the centroid of a set of points S is given by $(1/|S|)\Sigma_{p \in S} p$. In the sequel, a partition of a set S of n points into two sets of equal cardinality will be referred to as a bisection of S. The n points of an instance S of GC can be conveniently represented as the n columns of a $d \times n$ matrix B. Let $C = B^T B$ be considered as the connection matrix of a graph $G(V, E)$ on n vertices. We can consider an integer i to either be the i-th vertex in G or the i-th point in the set S. Therefore, there exist a one-to-one correspondence between a bisection of S and a bisection of G. In fact, we can consider a partition (X, Y) of the set $\{1, 2, \ldots, n\}$ into two parts X and Y of equal cardinality as a bisection of G or a bisection of S. The cost of a bisection (X, Y) of S is defined to be the negative of the Euclidean distance between the centroids of the points in X and Y.

Theorem 1: Let B a $d \times n$ matrix representing an even number n of points of a set S. Let $C = B^T B$ be considered as the connection matrix of a graph $G(V,E)$. Let c_1 and c_2 be the cost of a bisection (X,Y) in G and S respectively. Then, there exist two constants a and b independent of the bisection (X, Y) such that

$$c_2 = ac_1 + b. \qquad (**)$$

The relationship of MBP and GC given by Theorem 1 was first noted by Frankle and Karp [13]. However, a proof for Theorem 1 can also be found in [17]. As an immediate consequence of Theorem 1, a bisection (X, Y) is an optimal solution of GC if and only if it is an optimal solution of the corresponding instance of MBP. Thus if we can solve *GC* in polynomial time, then we can construct test cases for MBP with known optimal cost in polynomial time as follows:

Step 1. Generate a random $d \times n$ matrix, where n is even.

Step 2. Solve GC using the column of B as points in d-dimensional Euclidean space. Let c_2 be the cost of the optimal solution for GC.

Step 3. Construct the graph G given by its connection matrix $C = B^T B$. The cost of an optimal bisection in G is given by (**).

Let (X, Y) be a partition of a set S of points in a d-dimensional Euclidean space E, and let H be a hyperplane in E. We say that H *separates* (X, Y) if the set S can be split into three subsets L, R, and M, where L and R lie on either side of H, $M \subseteq H$, $L \subseteq X$, and $R \subseteq Y$. Furthermore, if $x, y \in M \Rightarrow x = y$, then we say that (X, Y) is *well separated* by H. The following two theorems which are proved in [18] form the basis for a polynomial-time algorithm for GC in any fixed dimension.

Theorem 2: Let S be a set of an even number of points in a Euclidean space E. If (X, Y) is an optimal bisection of S, then (X, Y) is well separated.

Theorem 3: Let S be a set of an even number of points in a Euclidean space E of dimension d such that the affine hull of S is E, and suppose that (X, Y) is a well separated partition of S. Then, there exists a sequence of affine subspaces $E = H_d \supseteq H_{d-1} \supseteq \cdots \supseteq H_1 \supseteq H_0$, where

a) $dim(H_k) = k$, $0 \leq k \leq d$.

b) For $1 \leq k \leq d$, H_k is the affine hull of $k + 1$ affinely independent points in S.

c) $X = \bigcup_{k=0}^{d} L_k$ and $Y = \bigcup_{k=0}^{d} R_k$, where, if $k > 0$, then L_k and R_k lie on either side of H_{k-1} in H_k, and L_0 and R_0 contain at most the single point (possibly repeated) that is in H_0.

Theorem 2 says that an optimal bisection must be well separated, and Theorem 3 gives us a procedure for enumerating well-separated partitions among which an optimal bisection exist. Thus, by going through all possible sequences of hyperplanes $H_d, H_{d-1}, \ldots, H_0$, one is bound to find an optimal solution for GC. In fact, this is exactly the algorithm of Montgomery-Smith and Saab. A straightforward analysis shows that this algorithm runs in $O(n^{O(d^2)})$ on n points in a d-dimensional space. However, for points in general positions, this algorithm can be expected to run in $O(n^{O(d)})$. Nevertheless, this algorithm remains highly exponential in the dimension d. Therefore, the method for constructing test cases for MBP using this algorithm for solving GC is only practical for small values of d. However, it is expected that test cases for MBP generated by the method of this section to be highly random.

6. CONCLUSION

In this paper, we have presented several methods for the generation of test cases for MBP for which the optimal cost is known. All the methods except the last one are based on the fact that MCP can be solved efficiently using network flow techniques. The last method uses a relationship between MBP and GC, and it relies on a polynomial-time algorithm for GC in any fixed dimension. The methods described

in this paper should be useful in the comparison of the various existing heuristic algorithms for MBP.

References

[1] W. Donath, "Logic Partitioning," *Physical Design Automation of VLSI Systems, B. Preas and M. Lorenzetti, Eds.* Menlo Park, CA: The Benjamin/Cummings Publishing Company, Inc., 1988.

[2] S. Goto and T. Matsuda, "Partitioning, Assignment and Placement," *Layout Design and Verification, T. Ohtsuki, Ed.* New York, NY: North-Holland, 1986.

[3] M.A. Breuer, "Min-Cut Placement," *Journal of Design Automation and Fault-Tolerant Computing*, vol. 1, no. 4, pp. 343–362, October 1977.

[4] A.E. Dunlop and B.W. Kernighan, "A Procedure for Placement of Standard-Cell VLSI Circuits," *IEEE Transactions on Computer-Aided Design*, vol. CAD-4, no. 1, pp. 92–98, January 1985.

[5] P. Suaris and G. Kedem, "An Algorithm for Quadrisection and its Application to Standard Cell Placement," *IEEE Transactions on Circuits and Systems*, vol. 35, no. 3, March 1988.

[6] Sheldon B. Akers, "Partitioning for Testability," *Journal of Design Automation and Fault-Tolerant Computing*, vol. 1, no. 2, pp. 133–146, February 1977.

[7] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY: W.H. Freeman and Company, 1979.

[8] B.W. Kernighan and S. Lin,"An Efficient Heuristic Procedure for Partitioning Graphs," *Bell System Technical Journal*, vol. 49, pp. 291–307, February 1970.

[9] E.R. Barnes, "An Algorithm for Partitioning the nodes of a Graph," *SIAM Journal of Algebraic and Discrete Methods*, vol. 3, no. 4, pp. 541–550, December 1982.

[10] T. Bui, S. Chaudhuri, T. Leighton, and M. Sipser, "Graph Bisection Algorithm with Good Average Case Behavior," *Proceedings of the 25th IEEE Symposium on Foundations of Computing*, pp. 181–192, 1984.

[11] C. Fiduccia and R. Mattheyses, "A Linear-Time Heuristics for Improving Network Partitions," *Proceedings of the 19th Design Automation Conference*, pp. 175–181, January 1982.

[12] Y. Saab and V. Rao, "Fast Effective Heuristics for the Graph Bisectioning Problem," *IEEE Transactions on Computer-Aided Design*, vol. 9, no. 1, pp. 91–98, January 1990.

[13] J. Frankle and R.M. Karp, "A Graph Partitioning Algorithm Based on Spectral Analysis," *Proceedings of SRC Tropical Research Conference*, University of California, Berkeley, February 1985.

[14] B. Krishnamurthy, "An Improved Min-Cut Algorithm for Partitioning VLSI Networks," IEEE Transactions on Computers, vol. C-33, no.. 5, pp. 438–446, May 1984.

[15] Y. Wei and C. Cheng, "A Two-Level Two-Way Partitioning Algorithm," Proceedings of the Conference on Computer-Aided Design, pp. 516–519, November 1990.

[16] S. Even, *Graph Algorithms*. Rockville, MD: Computer Science Press, Inc., 1979.

[17] Y. Saab, "Graph Bisectioning and Related Problems," *M.S. Thesis*, University of Illinois at Urbana-Champaign, January 1988.

[18] S.J. Montgomery-Smith and Y. Saab, "Geometric Clustering is Polynomial Time for Fixed Dimension," *Unpublished Manuscript, November 1990.*

[19] B. Krishnamurthy and P. Mellema, "On the Evaluation of Mincut Partitioning Algorithms for VLSI Networks," *Proceedings of the International Symposium on Circuits and Systems*, pp. 12–15, 1983.

[20] B. Krishnamurthy, "Constructing Test Cases for Partitioning Heuristics," *IEEE Transactions on Computers*, vol. C-36, no. 9, pp. 1112–1114, September 1987.

[21] V. Rao, *Private Communication*.

Biography

YOUSSEF G. SAAB received the B.S. degree in computer engineering and the M.S. and the Ph.D. degrees in electrical engineering from the University of Illinois at Urbana-Champaign, Urbana, IL, in 1986, 1988, and 1990, respectively. He is currently an assistant professor in the department of computer science at the University of Missouri-Columbia.

From January 1986 to July 1990, he was a research assistant at the Coordinated Science Laboratory, Urbana, IL. His research interests include computer-aided design and layout of VLSI circuits, combinatorial optimization, computational geometry, and graph algorithms.

Dr. Saab is a member of IEEE, ACM, Tau Beta Pi, Eta Kappa Nu, Phi Kappa Phi, and the Golden Key National Honor Society.