Algorithms for a Class of Min-Cut and Max-Cut Problem

Teofilo F. Gonzalez[1] and Toshio Murayama[2]

[1] Department of Computer Science, University of California,
Santa Barbara, CA 93106, USA
[2] Sony Corporation System LSI Group,
4-14-1 Asahi-cho Atsugi Kanagawa, 243 Japan

Abstract. The *k-Min-Cut (k-Max-Cut)* problem consists of partitioning the vertices of an edge weighted (undirected) graph into k sets so as to minimize (maximize) the sum of the weights of the edges joining vertices in different subsets. We concentrate on the k-Max-Cut and k-Min-Cut problems defined over complete graphs that satisfy the triangle inequality, as well as on d-dimensional graphs. For the one-dimensional version of our partitioning problems, we present efficient algorithms for their solution as well as lower bounds for the time required to find an optimal solution, and for the time required to verify that a solution is an optimal one. We also establish a bound for the objective function value of an optimal solution to the k-Min-Cut and k-Max-Cut problems whose graph satisfies the triangle inequality. The existence of this bound is important because it implies that any feasible solution is a near-optimal approximation to such versions of the k-Max-Cut and k-Min-Cut problems.

1 Introduction

Partitioning a set of objects into equal size subsets is a fundamental problem that arises in many disciplines including CAD (placement of devices) [9] and sparse matrix computations [5]. Our partition problems consist of an integer k, and an edge weighted undirected graph $G = (V, E, W)$, where V is a set of n vertices, E is the edge set, and W is the edge weight function $W : E \rightarrow \Re_0^+$ (the set of non-negative reals). The *k-Min-Cut (k-Max-Cut)* problem consists of partitioning V into k sets so as to minimize (maximize) the sum of the weights of the edges joining vertices in different subsets.

It is well known that the k-Min-Cut and the k-Max-Cut problems are NP-hard even when k is two ([3], and [4]). As a result of this, numerous researchers have studied heuristics, and restricted versions defined over special classes of graphs, e.g. planar graphs, interval graphs, mesh graphs, etc ([5], [6], [9], [10], and [12]). In this paper we concentrate on the k-Max-Cut and k-Min-Cut problems defined over complete graphs that satisfy the triangle inequality, as well as on a restricted class of these graphs called d-dimensional.

We say that a k-Max-Cut (k-Min-Cut) problem is one-dimensional, denoted by *(k,1)-Max-Cut ((k,1)-Min-Cut)*, if the graph is complete, and the set of vertices V can be placed along a straight line in such a way that the weight of each

edge is the distance between the two vertices it joins. The *(k,d)-Max-Cut* and *(k,d)-Min-Cut* are defined as the one-dimensional case, but the set of points lie in d-dimensional Euclidean space.

An instance of the k-Min-Cut and k-Max-Cut problems is said to *satisfy the triangle inequality* if the set of edges is complete, and the weight of an edge between any pair of vertices is never larger than the sum of the weights of the edges in any path between such vertices. We shall refer to the problems restricted this way as the *(k,t)-Min-Cut* and *(k,t)-Min-Cut* problems.

In [11] it was established that the (k,t)-Min-Cut and (k,t)-Max-Cut are NP-hard. The proofs are based on the reductions that show that simple Max-Cut is NP-hard [3]. The $(k,2)$-Max-Cut was also shown to be NP-hard in [11]. The proof is involved and follows an approach similar to the one used to establish that a 2-dimensional clustering problem is NP-hard [7].

In this paper we consider the one-dimensional version of our partitioning problems which in our notation are called the $(k,1)$-Max-Cut and $(k,1)$-Min-Cut problems. We present a characterization of optimal solutions to these problems, and present a simple $O(n\ log\ n)$ time algorithms to generate optimal solutions, where n is the number of vertices. Let k be any fixed constant. For the $(k,1)$-Max-Cut, we present a faster algorithm that takes only $O(n)$ time, and for the $(\frac{n}{k},1)$-Max-Cut problem we establish robust $\Omega(n\ log\ n)$ lower bound for the time required to compute an optimal partition. For the $(k,1)$-Min-Cut we discuss an $O(n)$ time algorithm, and for the $(\frac{n}{k},1)$-Min-Cut problem we show that verifying whether or not a solution is an optimal solution requires $\Omega(n\ log\ n)$ time. Of course the lower bounds hold only on certain models of computation.

We also establish a bound for the objective function value of an optimal solution to the (k,t)-Min-Cut and (k,t)-Max-Cut problems, for all k. We give problem instances, for all k, that nearly match the bounds when k is at least 5. For the case when k is two, we give problem instances that match the bound. The existence of these bounds is important because they imply that any feasible solution can be used to approximate the (k,t)-Max-Cut and (k,t)-Min-Cut, and such an approximation is near-optimal for reasonable values of k.

2 One-Dimensional Problems

In this section we study the $(k,1)$-Max-Cut and the $(k,1)$-Min-Cut problems. We present a characterization of optimal solutions to these problems, and present simple $O(n\ log\ n)$ time algorithms to generate optimal solutions. Let k be any fixed constant. For the $(k,1)$-Max-Cut, we present a faster algorithm that takes only $O(n)$ time, and for the $(\frac{n}{k},1)$-Max-Cut problem we establish an $\Omega(n\ log\ n)$ lower bound for the time required to compute an optimal partition. For the $(k,1)$-Min-Cut we discuss an $O(n)$ time algorithm, and for the $(\frac{n}{k},1)$-Min-Cut problem we show that verifying whether or not a solution is an optimal solution requires $\Omega(n\ log\ n)$ time. Of course the lower bounds hold only on certain models of computation.

Let $P_1, P_2, \cdots, P_{km-1}, P_{km}$ be the set of points on a straight line for an instance of the $(k,1)$-Max-Cut or $(k,1)$-Min-Cut problems, where $km = n$. We assume without loss of generality that $P_1 \leq P_2 \leq \cdots \leq P_{km-1} \leq P_{km}$. A characterization of optimal solutions to the $(k,1)$-Min-Cut and $(k,1)$-Max-Cut problems is given by the following theorem.

Theorem 1. *The partition* $\{S_1, S_2, \cdots, S_k\}$ $(\{S'_1, S'_2, \cdots, S'_k\})$ *is an optimal solution to the* $(k,1)$*-Min-Cut* ($(k,1)$*-Max-Cut*) *problem, where*

$$S_i = \{P_i, P_{i+k}, \cdots, P_{i+k(m-2)}, P_{i+k(m-1)}\} \text{ for } 1 \leq i \leq k, \text{ and}$$

$$S'_i = \{P_{(i-1)m+1}, P_{(i-1)m+2}, \cdots, P_{(i-1)m+m-1}, P_{im}\} \text{ for } 1 \leq i \leq k.$$

Proof. We prove this theorem by showing that a partition with smaller (larger) objective function value can be obtained for any partition that differs from the S (S') partition. This is established through the use of interchange arguments. For brevity the proof is omitted.

One can easily show that the partition $\{S'_1, S'_2, \cdots, S'_k\}$ given in Theorem 1 is the only optimal solution to the $(k,1)$-Max-Cut problem. As pointed out by an anonymous referee, the partition $\{S_1, S_2, \cdots, S_k\}$ is not the only optimal solution to the $(k,1)$-Min-Cut problem. In this case one can show that any partition in which the points $P_{(i-1)m+1}, P_{(i-1)m+2}, \cdots, P_{(i-1)m+m-1}, P_{im}$, for $1 \leq i \leq k$ belong to distinct sets is an optimal solution. One can show that this represents the sets of all optimal solutions to the $(k,1)$-Min-Cut problem. Let us now discuss algorithms that generate optimal solutions based on the above characterizations.

Algorithm *sort-it-Min* for the (k,1)-Min-Cut problem reads in the set of points and then sorts them from smallest to largest. The points are then traversed in that order and are assigned to the sets $S_1, S_2, \cdots, S_k, S_1, S_2, \cdots, S_k$, and so on. Clearly, this procedure takes $O(n \log n)$ time. Similarly, algorithm *sort-it-Max* for the (k,1)-Max-Cut problem reads in the points and sorts them from smallest to largest. The smallest m points are assigned to S'_1, the next smallest m points are assigned to S'_2, and so on. This procedure also takes $O(n \log n)$ time. We formalize these results in the following theorem that we state without a proof.

Theorem 2. *Algorithm sort-it-Min (sort-it-Max) solves in O(n log n) time the (k,1)-Min-Cut ((k,1)-Max-Cut) problem.*

An alternate algorithm, which we call *2-fast-Max*, for the $(2,1)$-Max-Cut problem first *selects* the smallest $\frac{n}{2}$ points, and assigns them to S'_1. The remaining points are assigned to S'_2. This algorithm takes $O(n)$ time because selection takes linear time [8]. A similar algorithm, *k-fast-Max*, for the $(k,1)$-Max-Cut takes linear time. Note that k in this case is not an input to the algorithm, i.e., k is a fixed constant in the algorithm (but n is a variable). Thus, we have the following theorem.

Theorem 3. *Algorithm k-fast-Max solves n O(n) time the (k,1)-Max-Cut problem, for any fixed value of k.*

The question now is whether or not a linear time algorithm exists for the other cases? In what follows we show that linear time decision tree algorithms do not exist for the other one dimensional versions of the problem. More specifically, we establish an $\Omega(n \log n)$ lower bound for the $(\frac{n}{2}, 1)$-Max-Cut problems.

By theorem 1 we know that an optimal solution to this problem is of the following form: $\{P_1, P_2\}, \{P_3, P_4\}, \ldots, \{P_{n-1}, P_n\}$. We now show that any algorithm that finds an optimal solution to the $(\frac{n}{2}, 1)$-Max-Cut problem in $T(n)$ time can be used to determine element uniqueness ([2]) of $\frac{n}{2}$ real numbers in $T(n)+O(n)$ time. Such algorithm works as follows. Let $X = (x_1, x_2, \ldots, x_{\frac{n}{2}})$ be the $\frac{n}{2}$ elements in the element uniqueness problem. Let min (max) the smallest number in X. There are two sets of inputs to the $(\frac{n}{2}, 1)$-Max-Cut problem. The first input set is X, and the second input set is X plus the numbers $min - 1$ and $max + 1$. The output to the element uniqueness problem is "yes" if, and only if, the two elements in each of the sets that form an optimal solution to both of the instances of the $(\frac{n}{2}, 1)$-Max-Cut are distinct. Since this checking can be done in $O(n)$ time, we know that any algorithm that solves the $(\frac{n}{2}, 1)$-Max-Cut problem in $T(n)$ time can be used to determine element uniqueness for a set of $\frac{n}{2}$ elements in $T(n) + O(n)$ time. As a result of this, "any" lower bound for element uniqueness also holds for $(\frac{n}{2}, 1)$-Min-Cut problem. Since an $\Omega(n \log n)$ lower bound for element uniqueness is known ([1], and [2]), we also have an $\Omega(n \log n)$ for the $(\frac{n}{2}, 1)$-Min-Cut problem. The following theorem is a trivial extension of these observation.

Theorem 4. *Any algorithm that solves $(\frac{n}{k}, 1)$-Max-Cut problem, for any fixed value of k, in $T(n)$ time can be used to determine element uniqueness for a set of $\frac{n}{2}$ elements in $T(n) + O(n)$ time.*

Let us now discuss very nice $O(n)$ time algorithm for the $(\frac{n}{2}, 1)$-Min-Cut problem which was developed by an anonymous referee. By the comments just after theorem 1 we know that an optimal solution to the $(\frac{n}{2}, 1)$-Min-Cut problem is of the following form: each of the $\frac{n}{2}$ sets contains exactly two points, one from $\{P_1, P_2, \ldots, P_{\frac{n}{2}}\}$ and the other from the remaining points. This characterization suggests a simple $O(n)$ algorithm to construct an optimal partition. First we find the median using the linear time algorithm in [8]. Then we assign each of the first $\frac{n}{2}$ points to distinct sets, the remaining points are assigned similarly. It is simple to see that this linear time algorithm generates an optimal solution to the $(\frac{n}{2}, 1)$-Min-Cut problem, and that it can be easily extended to solve the $(\frac{n}{k}, 1)$-Min-Cut problem, for any fixed value for k.

For the $(2, 1)$-Min-Cut we do not know of any $O(n)$ for its solution, and we do not know of any nontrivial lower bound for it. However, we feel an $\Omega(n \log n)$ lower bound is likely to exist. We base this on the following results for the verification version of these problems. By the verification version of a problem we mean given a problem and a solution, is the solution an optimal solution to

the problem. In this case the solution is a partition and we are asked to decide whether or not the partition is an optimal one for the problem.

The $(2,1)$-Max-Cut verification problem can be easily solved in $O(n)$ time by just checking whether or nor all the points in one set of the partition are either located to the left, or to the right, of all the points in the other set. A similar algorithm can be developed for the $(k,1)$-Max-Cut verification problem. The $(\frac{n}{2},1)$-Min-Cut verification problem is a little bit more complex. The problem can be reduced to checking whether or not the sum of the distance between each pair of points in each set in the given solution is equal to the corresponding sum of the sets in an optimal solution, which we know can be computed in $O(n)$ time. A similar algorithm can be developed for the $(\frac{n}{k},1)$-Min-Cut verification problem, for any constant k.

Theorem 5. *The $(\frac{n}{k},1)$-Min-Cut and the $(k,1)$-Min-Cut verification problems, for any fixed value for k, can be solved in $O(n)$ time by the above algorithms.*

Proof. By the above discussion.

For the $(\frac{n}{2},1)$-Max-Cut and the $(2,1)$-Min-Cut verification problems we establish an $\Omega(n \log n)$ lower bound in what follows. Before presenting those results, we need to define a decision problem related to the element uniqueness problem. The *element ϵ-uniqueness* decision problem consists of $\frac{n}{2}$ real numbers, and a real number $\epsilon > 0$. An instance is a yes-instance if, and only if, every two of such numbers are at least ϵ units from each other. It is simple to show that "any" lower bound that can be established for the element uniqueness problem can also be established for the element ϵ-uniqueness problem as long as the bound is derived strictly for the number of "yes" components.

We now show that any algorithm that solves the $(\frac{n}{2},1)$-Max-Cut verification problem in $T(n)$ time can be used to determine element ϵ-uniqueness of $\frac{n}{2}$ real numbers in $T(n)+O(n)$ time. The resuction is defined as follows. Let $X = (x_1, x_2, \ldots, x_{\frac{n}{2}})$ be the $\frac{n}{2}$ elements in the element ϵ-uniqueness problem. The set of inputs to the $(\frac{n}{2},1)$-Max-Cut verification problem is X followed by $x_1+\epsilon, x_2+\epsilon, \ldots, x_{\frac{n}{2}}+\epsilon$, and the partition which we want to determine whether or not is an optimal one is: $\{x_1, x_1+\epsilon\}, \{x_2, x_2+\epsilon\}, \ldots, \{x_{\frac{n}{2}}, x_{\frac{n}{2}}+\epsilon\}$. With this linear time reduction it is simple to establish the following theorem.

Theorem 6. *Any algorithm that solves $(\frac{n}{k},1)$-Max-Cut verification problem, for any fixed value for k, in $T(n)$ time can be used to determine element ϵ-uniqueness for a set of $\frac{n}{2}$ elements in $T(n)+O(n)$ time.*

Proof. The proof is based on the above reduction.

We now show that any algorithm that solves the $(2,1)$-Min-Cut verification problem in $T(n)$ time can be used to determine element ϵ-uniqueness of $\frac{n}{2}$ real numbers in $T(n)+O(n)$ time. The reduction is defined as follows. Let $X = (x_1, x_2, \ldots, x_{\frac{n}{2}})$ be the $\frac{n}{2}$ elements in the element ϵ-uniqueness problem. The set of inputs to the $(2,1)$-Min-Cut verification problem is X followed by x_1+

$\epsilon, x_2 + \epsilon, \ldots, x_{\frac{n}{2}} + \epsilon$, and the partition which we want to determine whether or not it is an optimal one is: $\{x_1, x_2, \ldots, x_{\frac{n}{2}}\}$ and $\{x_1 + \epsilon, x_2 + \epsilon, \ldots, x_{\frac{n}{2}} + \epsilon\}$. After extending this linear time reduction for any fixed value for k, we can easily establish the following theorem.

Theorem 7. *Any algorithm that solves $(k,1)$-Min-Cut verification problem, or any fixed value for k, in $T(n)$ time can be used to determine element ϵ-uniqueness for a set of $\frac{n}{2}$ elements in $T(n) + O(n)$ time.*

Proof. The proof is based on the above reduction.

3 Triangle Inequality Case

Let us now consider the (k,t)-Min-Cut and (k,t)-Max-Cut problems, i.e., the graph satisfies the triangle inequality. In this section we establish a bound between $f^*_{(k,t)-Min}(I)$ and $f^*_{(k,t)-Max}(I)$, where $f^*_{(k,t)-Min}(I)$ and $f^*_{(k,t)-Max}(I)$ are the objective function values of optimal solutions to the (k,t)-Min-Cut and (k,t)-Max-Cut, respectively.

Theorem 8. *Let I be any instance of the (k,t)-Min-Cut ((k,t)-Max-Cut) problem. Then,*

$$f^*_{(k,t)-Max}(I) \leq \frac{k+1}{k-1} f^*_{(k,t)-Min}(I), \text{ and } f^*_{(2,t)-Max}(I) \leq 2 f^*_{(2,t)-Min}(I).$$

Proof. The proofs are based on elaborate accounting of the cost of transforming from one solution to the other. For brevity the proofs are omitted.

We now show that $f^*_{(2,1)-Max}$ can be arbitrarily close to $2f^*_{(2,1)-Min}$. Consider the example given in Fig. 1. From Theorem 1, we know that an optimal $(2,1)$-Min-Cut partition is

$$S_1 = \{x_1, x_3\} \text{ and } S_2 = \{x_2, x_4\},$$

and an optimal $(2,1)$-Max-Cut partition is

$$S'_1 = \{x_1, x_2\} \text{ and } S'_2 = \{x_3, x_4\}.$$

Then, we know that

$$\lim_{d \to 0} \frac{f^*_{(2,1)-Max}(I)}{f^*_{(2,1)-Min}(I)} = 2.$$

If two vertices are allowed to be at the same location, the above bound becomes exactly 2.

From theorem 2.1 we know that any algorithm which partitions the input into two equal-sized subsets and generates a solution with objective function value $f(I)$ has the property that

$$\frac{|f^*_{(2,1)-Min} - f(I)|}{f^*_{(2,1)-Min}} \leq 1, \quad \text{and} \quad \frac{|f^*_{(2,1)-Max} - f(I)|}{f^*_{(2,1)-Max}} \leq \frac{1}{2}.$$

Fig. 1. Example where $f^*_{(2,1)-Max}(I) \approx 2f^*_{(2,1)-Min}(I)$. The distance d is very small.

Thus, any algorithm is a 1-approximation algorithm for the $(2,1)$-Min-Cut problem and a $\frac{1}{2}$-approximation algorithm for the $(2,1)$-Max-Cut problem when the input graph satisfies the triangle inequality. These results suggest that there may be other approximation algorithms with a smaller approximation bound.

We know establish that for any positive integer k (≥ 2), there exists an instance I such that

$$f^*_{(k,t)-Max}(I) = \frac{k}{k-1} f^*_{(k,t)-Min}(I).$$

There are nk points partitioned into k subsets $\{S_1, S_2, \cdots, S_k\}$. All the points in each subset have the same coordinate value. Points in different subsets are at a distance d from each other. In this case, the partitions T_1 and T_2 satisfy

$$\frac{t_1}{t_2} = \frac{k}{k-1}$$

where T_1 is the partition $\{S'_1, S'_2, \cdots, S'_k\}$ and T_2 is the partition $\{S_1, S_2, \cdots, S_k\}$, and t_1 and t_2 are the objective function values of T_1 and T_2, respectively. It is simple to show that T_1 is an optimal (k,t)-Max-Cut partition, and T_2 is an optimal (k,t)-Min-Cut partition (the proof is similar to Theorem 1). Therefore, we know that

$$\frac{f^*_{(k,t)-Max}(I)}{f^*_{(k,t)-Min}(I)} = \frac{k}{k-1}.$$

We know from this result that when k is large, the objective function value of any partition is near optimal regardless of n.

4 Discussion

We presented a characterization of optimal solutions to the $(k,1)$-Max-Cut and the $(k,1)$-Min-Cut problems. This characterization enabled us to develop a simple $O(n\ log\ n)$ time algorithms to generate optimal solutions. Let k be any fixed constant. For the $(k,1)$-Max-Cut, we presented a faster algorithm that takes only

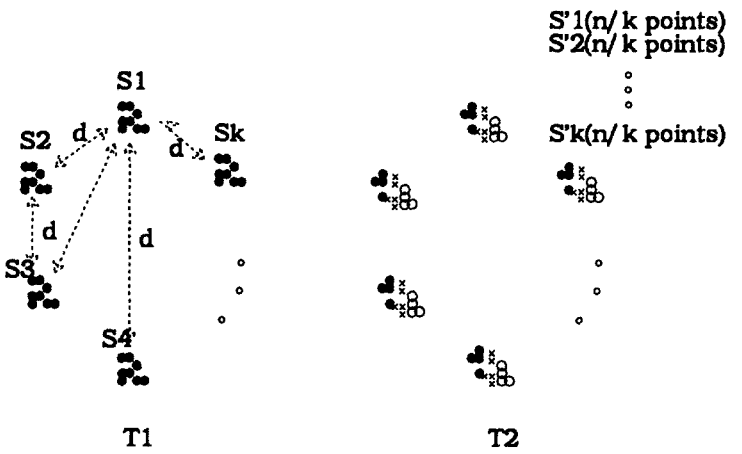

Fig. 2. Possible Upper Bound. T1 (T2) is an optimal solution for the (k,t)-Max-Cut ((k,t)-Min-Cut) problem.

$O(n)$ time, and for the $(\frac{n}{k},1)$-Max-Cut problem we established robust $\Omega(n\ log\ n)$ lower bound for the time required to compute an optimal partition. For the $(k,1)$-Min-Cut we discussed an $O(n)$ time algorithm developed by an anonymous referee, and for the $(\frac{n}{k},1)$-Min-Cut problem we showed that verifying whether or not a solution is an optimal solution requires $\Omega(n\ log\ n)$ time.

We also established a bound for the objective function value of an optimal solution to the (k,t)-Min-Cut and (k,t)-Max-Cut problems. We presented problem instances, for all k, that nearly match the bounds when k is at least 5. For the case when k is two, we presented a problem instance that match the bound. The existence of these bounds is important because they imply that any feasible solution can be used to approximate the (k,t)-Max-Cut and (k,t)-Min-Cut, and such an approximation is near-optimal for reasonable values of k. For a special type of problem instances we can establish tight bounds. For brevity we did not discuss these results.

Several interesting heuristics for our 2-dimensional partitioning problems were developed, and results of an experimental evaluation of these methods is given in [11]. A very interesting open problem is that of finding efficient approximation algorithms for the two dimensional case. A very interesting problem is to determine whether or not the (2,2)-Max-Cut problem is an NP-hard problem. In [11] it was shown that there are problem instances of the (2,2)-Max-Cut problem which do not have an optimal solution in which the two sets in the partition can be separated by k lines, for any fixed value of k.

References

1. M. Ben-Or, "Lower Bounds for Algebraic Computation Trees," Proc. 15^{th} ACM Annual Symposium on Theory of Computing 80 - 86, 1983.
2. D. Dobkin, and R. Lipton, "Multidimensional Searching Problems," *SIAM Journal on Computing*, 15(2), 181 - 186, 1976.
3. M. R. Garey, D. S. Johnson, and L. Stockmeyer, "Some Simplified NP-Complete Graph Problems," *Theoretical Computer Science*, (1), 237 - 267, 1976.
4. M. R. Garey and D. S. Johnson, *"Computers and Intractability,"* Freeman, 1980.
5. A. George and J. W. H. Liu, "An automatic nested dissection algorithm for irregular finite element problems," *SIAM Journal on Numerical Analysis*, 15, 1053 - 1069, 1978.
6. J. R. Gilbert and E. Zmijewski, *"Combinatorial Sparse Matrix Theory,"* Chapter 5, Lecture Notes in Computer Science, UCSB, 1991.
7. T. F. Gonzalez, "Clustering to Minimize the Maximum Intercluster Distance," *Theoretical Computer Science*, 38, 293 - 306, 1985.
8. E. Horowitz and S. Sahni, *"Fundamentals of Computer Algorithms,"* Computer Science Press, 1978.
9. B. W. Kernighan and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs," *The Bell System Technical Journal*, 49, 291 - 307, 1970.
10. B. Krishnamurthy, "An improved min-cut algorithm for partition VLSI networks," *IEEE Transactions on Computers*, C(33), 438 - 446, 1984.
11. T. Murayama, "Algorithmic Issues for the Min-Cut and Max-Cut Problems," M.S. Thesis, Department of Computer Science, The University of California, Santa Barbara, December 1991.
12. D. F. Wong, H. W. Leong, and C. L. Liu, *"Simulated Annealing for VLSI Design,"* Kluwer Academic Publishers, 1988.