

Multi-way Partitioning Using Bi-partition Heuristics

Maogang Wang

ECE Department
Northwestern University
Evanston, IL, US
mgwang@ece.nwu.edu

Sung Kyu Lim

CS Department
UCLA
Los Angeles, CA, US
limsk@cs.ucla.edu

Jason Cong

CS Department
UCLA
Los Angeles, CA, US
cong@cs.ucla.edu

Majid Sarrafzadeh

ECE Department
Northwestern University
Evanston, IL, US
majid@ece.nwu.edu

Abstract— The multi-way partition problem is very important in various applications. In this paper, we use analytical and experimental results to study the k -way partition problem. We introduce the concept of embedding graph for the the k -way partition problem. Based on this concept, we explain different scenarios of using a bi-partition heuristic to solve the k -way partition problem. If C denote the optimal cut cost for the k -way partition problem and the bi-partition heuristics we use are δ -approximation heuristics (defined in Section 2), we prove that the cut cost from the hierarchical approach has an approximate upper bound of $\delta C \cdot \log k$ while the cut cost from the all-way bi-partition, or flat approach, has an upper bound of $\delta C k$. This is contrary to some claims made in recent literature (and CAD tools designed based on it). Experimental results strongly support our theoretical analysis. Our results show that for large target graph, the hierarchical approach is about 77% better than the single-pass all-way bi-partition approach. The all-way bi-partition approach will perform better in a multi-pass set-up. However, the hierarchical approach is still on average 7.1% better in quality and 144 times faster than the multi-partition all-way bi-partition approach. The main conclusion of this paper is, contradicted to what has been suggested in literature, hierarchical bi-partitioning is a more effective multi-way partitioning scheme.

I. INTRODUCTION

Graph partitioning is an important problem and has extensive application to many areas in VLSI design [3]. The problem is to partition the vertices of a graph in k into roughly equal parts, such that the number of edges connecting vertices in different parts is minimized. In this paper, to simply the presentation, we use a graph model. All analysis is extensible to a hypergraph model. Formally, a graph $G = (V, E)$ is defined as a set of vertices V and a set of edges E . Most existing researches are focused on the bi-partition problem. A high quality graph partitioning algorithm greatly affects the feasibility, quality, and cost of the resulting system.

The graph partition problem is NP-complete. Recently,

a number of researchers have investigated a class of algorithms that can give a reasonably good solution for the bi-partition problem [14, 9, 4, 10, 12, 13]. However, algorithm designers are more and more interested in the general k -way partition problem where k is greater than two.

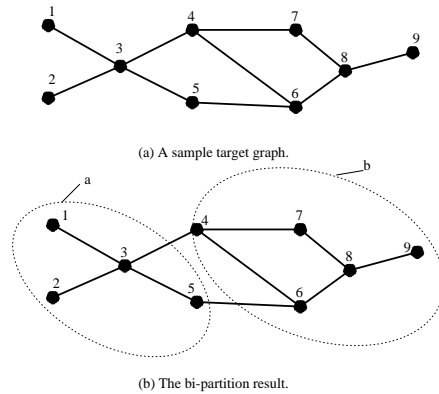


Fig. 1. An example of the k -way partition problem.

Figure 1a shows a sample graph to be partitioned. We call this a target graph. We call all the vertices in the target graph nodes, and edges in the target graph links. When $k = 2$, the bi-partition result is shown in Figure 1b. The cut cost for bi-partition shown is 2. Figure 2a shows a 3-way partition result where the cut cost is 4. We can also perform a 4-way partitioning on this graph. Figure 2b shows the result with the cost being 5.

When k is greater than 2, there are three typical methods to solve the k -way partition problem.

Method A is a direct extension of 2-way FM-like algorithms. In the 2-way FM-like algorithms, each node can be moved to only one definite destination partition. A gain will be associated to this move. In the k -way partition problem, each node has $k - 1$ possible destination partitions. Thus method A is based on the 2-way FM algorithm while allowing moving any node to any of the $k - 1$ partitions. Method B is a all-way bi-partition improvement. It starts with an initial (and arbitrary) k -way

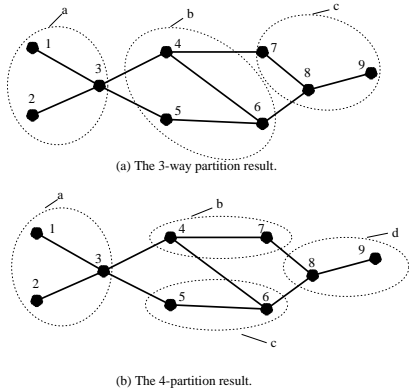


Fig. 2. An example of the k -way partition problem.

partition. It picks two partitions from the total k partitions at a time and perform bi-partitioning to improve the all-way cut-cost between these two partitions. Method C is a hierarchical approach. It will recursively bi-partition the target graph until we have k partitions.

Suaris and Kedem used method A on a 4-way partition problem [18, 19]. For k greater than 4, this method A is rarely used since it needs a lot of memory and is slow. In [7], Cong and Lim show that method A produces almost 500% worse than method C, the hierarchical approach, in large graph. Given the fact that there are a number of high-quality bi-partition heuristics/codes [1, 6, 8, 12] and that multi-way partition method A is not satisfactory, we should find a method of solving the k -way partition problem using a bi-partition heuristic. In practice, the all-way bi-partition approach (Method B) and the hierarchical approach (Method C) are often used. The question is that which one is a better way to solve the k -way partition problem. In the original Kernighan-Lin’s paper [14], the author argues that the hierarchical approach is a very greedy method. Intuitively, the hierarchical approach can not recover the possible mistakes made by the previous cuts. Thus it will easily get stuck into a local minimum. They also argue that given enough time, the all-way bi-partition method should be able to explore most part of the solution space. Thus it has a better chance to get a good result. In [7], Cong and Lim experimentally show that an all-way bi-partition approach can produce about 10% better than the hierarchical approach. However, their all-way bi-partition algorithm uses the partition result from a hierarchical algorithm as the input partition. Obviously, this is not an comparison between the hierarchical and the all-way approach. We will compare these two approaches here. More work on k -way partition can be found in [16, 5, 20, 2, 15, 17].

In this paper, we use analytical and experimental results to study the k -way partition problem. We introduce the concept of embedding graph for the k -way partition problem. Based on this concept, we explain different scenarios of using a bi-partition heuristic to solve the k -way

partition problem. Theoretical analysis is done on the all-way bi-partition approach and the hierarchical approach assuming we use a δ -approximation bi-partition heuristic. If the optimal cut cost for a k -way partition problem is C , we prove that the cut cost from the hierarchical approach has an upper bound of $\delta C \cdot \log k$ while the cut cost from the all-way bi-partition approach has an upper bound of $\delta C k$ where k is the number of partitions. In [17], Simon and Teng proved that the hierarchical approach has an upper bound of $\delta C_0 \cdot \log k$ where C_0 is the optimal cut cost for a perfectly balanced k -way partition. Our experimental results support the analysis and reveals that the hierarchical approach is indeed a better way to solve the k -way partition problem than the all-way bi-partition approach. Specifically, on average, the hierarchical approach produces cut cost 7% better and is 144 times faster than the all-way bi-partition approach. In the experimental section, we also study the effect of the balancing criterion between partitions on the final partition result.

This paper is organized as follows: In Section 2 we formalize the k -way partition problem and introduce the concept of embedding graph for the k -way partition problem. In Section 3 we give theoretical upper bounds on various k -way partitioning algorithms. In Section 4 we show the experimental results to support our claim followed by the conclusion in Section 5.

We removed all the theoretical proofs in Section 3 due to the page limit. All proofs can be found in the original technical report.

II. USING BI-PARTITION HEURISTICS TO SOLVE THE k -WAY PARTITION PROBLEM

The bi-partition problem is a well-studied problem. There are a number of good algorithms out there to solve this problem. Alpert-Kahng and Hauck-Borriello gave a very detailed overview and comparison on various algorithms [3, 11]. Most of these algorithms are based on the KL-FM algorithm. The KL (Kernighan-Lin) algorithm uses a pair-swap move structure [14]. During each pass, every node is moved exactly once between the two partitions. At the beginning of the pass, all nodes are “unlocked”, i.e., free to be swapped. After the selected nodes are swapped, they become “locked” and the algorithm updates both the cost of the new partition and the gains of the remaining unlocked nodes. After all nodes are locked, the lowest-cost partition encountered over the entire pass is restored and returned. Further passes are executed, each using the result from the previous pass as its starting point, until no improvement results are obtained. Computing gains in the KL heuristic is expensive; $O(n^2)$ swaps are evaluated before every move, resulting in a complexity per pass of $O(n^2 \log n)$ (assuming a sorted list of costs). The FM (Fiduccia-Mattheyses) algorithm reduces the time per pass to linear in the size of the graph (i.e., $O(t)$, where t is the number of hyperedge end points

or terminals) by adopting a single-node move structure, and a *gain bucket* data structure that allows constant-time selection of the highest-gain node and fast gain updates after each move.

The FM algorithm can be easily extended to solve the k -way partition problem. A direct extension of the bi-partition FM algorithm is to allow each node move to any of the other $k - 1$ partitions, where the gain of each node should be the maximum gain among all these possible $k - 1$ moves. When updating the gain for a node, all these possible $k - 1$ moves need to be revisited. Thus the gain updates need $k - 1$ more time than in the bi-partition FM algorithm. Approximately, this method is computationally $k - 1$ times more expensive than the original bi-partition FM algorithm. Due to this fact, this method is rarely used when k is larger than four. In a VLSI circuit design problem, a target graph can easily have as many as a couple of hundred thousand nodes and up to several millions of nodes. The direct extension of the FM algorithm is too slow for such problems. We need a heuristic which is faster than this direct FM extension to solve the k -way partition problem.

In the graph partitioning problem, edge-cut is the objective to minimize. This objective can also be viewed as an embedding cost. Figure 3 demonstrates this fact. Figure 3a shows a 4-way partitioning scheme. We label these four partitions as a , b , c and d , respectively. According to the definition of edge-cut objective, an edge which connects vertices in two different partitions will have a cost of 1. We can embed this cost into a complete graph shown in Figure 3b. The cost for an edge which connects vertices in two partitions is the distance of the shortest path between these two partitions. For a complete graph, the distance between any two vertices is always 1. Therefore an embedding cost for a complete graph is identical to the edge-cut cost in the corresponding k -way partition problem. Figure 3c shows an example of embedding the target graph in Figure 2b into the embedding graph. Nodes 1, 2 and 3 are embedded in the partition/embedding vertex a , nodes 4, 7 are in the embedding vertex b , nodes 5, 6 are in the embedding vertex c and nodes 8, 9 are in the embedding vertex d . There are a link between the embedding vertex a and b , a and c , c and d , b and c , b and d , shown as the dotted line between partitions in Figure 3c. The total cut cost is 5.

Sometimes a k -way partition algorithm works only on a sub-graph of the original embedding graph. In this case the edge-cut cost will be the embedding cost for this sub-graph. We say that this algorithm is “looking at” n_e edges at this particular time, where n_e is the number of embedding edges of the sub-embedding-graph. For example, the direct FM extension looks at all $k(k - 1)/2$ embedding edges. The all-way bi-partition algorithm looks at only one embedding edge at a time.

In the original Kernighan-Lin’s paper, the authors have a low opinion of the hierarchical algorithms: “Obviously

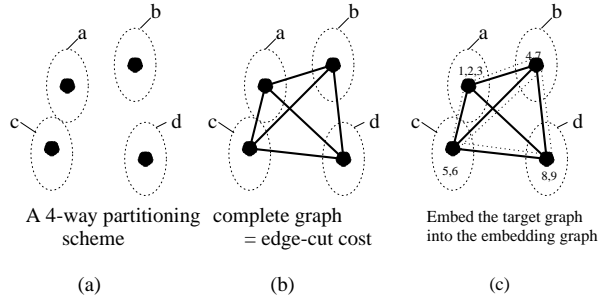


Fig. 3. Edge-cut cost is an embedding cost.

this (the first cut in the hierarchical approach) may conflict directly with the next stage, which is to try to divide each subset further. Carried to several levels, it can lead to a relatively poor overall solution.” We have also asked a number of researchers in this area. They all think the same. We will theoretically analyze the hierarchical and the all-way algorithm in the Section 3 and provide experimental results in Section 4.

III. THEORETICAL ANALYSIS

In this section we will give a theoretical analysis of different algorithms for solving the k -way partition problem using a bi-partitioning heuristic.

In order to make the result more general, the analysis should not be based on any specific bi-partition heuristic. Thus, we abstract any bi-partition heuristic to a so called δ -approximation algorithm: that is, the cost of the bi-partition found by this algorithm is no more than δ times the cost of the optimal bi-partition result.

In [17], Simon and Teng theoretically studied the hierarchical bi-partition approach in the k -way partition problem. The results showed that if the number of vertices in each partition is required to be exactly n/k (a perfectly balanced k -way partition), the hierarchical approach may get very bad results. However, if the balance condition is relaxed so that the number of vertices in each partition is bounded by $2n/k$, the hierarchical bi-partition approach will find an approximately balanced k -way partition whose cost is within an $O(\log k)$ factor of the cost of the optimal perfectly balanced k -way partition.

In practical applications, usually a perfectly balanced k -way partition is not necessary. A k -way partition heuristic is intended to find the optimal result for an approximately balanced k -way partition. Simon and Teng’s upper bound for the hierarchical approach is a factor of the optimal result of a perfectly balanced k -way partition. In this paper, we will show upper bounds for both hierarchical and all-way approach which is related to the optimal result of a relaxed k -way partition.

As the first result, we will show that the cost of the k -way partition found by the hierarchical approach is no more than $O(\delta \log k)$ times the optimal cost for this k -

way partition problem if each cut in the hierarchy is a 1/3-balanced cut.

Lemma 1 *In the hierarchical approach, the cost of the first cut is no more than δ times the optimal cost for the k -way partition problem.*

Lemma 2 *In the hierarchical approach, the total cost of the second level cuts is no more than δ times the optimal cost for the k -way partition problem.*

Lemma 3 *In the hierarchical approach, the total cost of cuts at any level is no more than δ times the optimal cost for the k -way partition problem.*

Theorem III.1 *The cost of the k -way partition found by the hierarchical approach is no more than $O(\delta \log k)$ times the optimal cost of the k -way partition problem.*

(All proofs are omitted here due to the page limit. Please contact the authors for a complete manuscript.)

This is an upper bound for the hierarchical approach. The next result we will provide is a similar upper bound for the all-way bi-partition approach.

Theorem III.2 *The cost of the k -way partition found by the all-way bi-partition approach is no more than $O(\delta(k-1))$ times the optimal cost for the k -way partition problem.*

As we stated before, this is an upper bound. It is possible to improve upper bounds for both approaches. However it is interesting to notice the difference between the two upper bounds ($\log k$ vs. $(k-1)$) both using the “same” analysis method. This difference suggests that the hierarchical approach is a better way to do the k -way partition problem.

IV. EXPERIMENTAL RESULTS

We experimentally evaluated the two methods we analyzed in the previous section, the hierarchical approach and the all-way bi-partition approach. We use ten MCNC standard-cell benchmark circuits. We choose the value of k to be 16 in this set of experiments.

Since hMetis [12, 13] is currently the best partitioner, we pick hMetis [12] as our basic bi-partition tool. However, the same results should hold for any other bi-partition algorithms, e.g., [9, 1, 6] (since our proofs/analysis/claims are independent of the bi-partition heuristic).

In the all-way bi-partition approach, we can have different ways to pick two partitions at a time to perform the bi-partition. The following four ways are discussed in [7]:

1. *Random*: randomly pick two partitions at a time.
2. *Exhaustive*: Use a specific sequence to exhaustively pick all possible pairs of partitions.

| circuit | one-pass exhaust. | | hierarchical | | % imp. |
|---------|-------------------|---------|--------------|---------|--------|
| | net-cut | time(s) | net-cut | time(s) | |
| fract | 74 | 5.2 | 56 | 1.21 | 24.3% |
| struct | 996 | 24.5 | 128 | 5.74 | 87.1% |
| p1 | 398 | 32.8 | 150 | 5.17 | 62.3% |
| p2 | 1469 | 26.2 | 450 | 12.5 | 69.4% |
| biomed | 2423 | 31.3 | 213 | 13.18 | 91.2% |
| in1 | 978 | 24.7 | 175 | 9.68 | 82.1% |
| in2 | 5257 | 72.1 | 898 | 44.9 | 82.9% |
| in3 | 10705 | 104 | 2070 | 80.1 | 80.7% |
| avqs | 9578 | 75.8 | 532 | 42.6 | 94.4% |
| avql | 10594 | 85.3 | 543 | 47.8 | 94.9% |
| avg. | | | | | 76.9% |

TABLE I
SINGLE-PASS ALL-WAY ALGORITHM VS. THE HIERARCHICAL ALGORITHM (CONCLUSION: THE HIERARCHICAL APPROACH IS MORE EFFECTIVE THAN THE ONE-PASS ALL-WAY APPROACH).

3. *Cut-based*: Pick a pair of two most tightly or loosely connected partitions, measure in terms of cutsizes.
4. *Gain-based*: Pick a pair of two partitions between which the cutsizes reduction is maximum or minimum during last pass.

In the all-way bi-partition scenario, we can always take an iterative approach which is to perform all-way bi-partitioning multiple *passes*. A pass is done when we finish looking at all the embedding edges or vertices in the embedding graph. The iterative procedure stops when the net-cut cost cannot be improved. However, the theoretical analysis we performed in Section 3 is truly based on looking at one embedding edge only *once*. This is equivalent to a single-pass all-way algorithm. We know that a multi-pass algorithm would definitely improve the results over the single-pass algorithm. Therefore, in order to justify the claims we made in Section 3, we compare the partition results of a single-pass all-way bi-partition algorithm to the results of a hierarchical algorithm. Since the gain-based all-way approach needs information from the previous pass, it cannot be used as a single-pass algorithm. Among the remaining three methods, we empirically found that the exhaustive method performs the best. Table I shows the partition results from an exhaustive single-pass all-way approach and from a hierarchical approach.

For all circuits, the hierarchical approach is remarkably better than the single-pass all-way approach. The average improvement of the hierarchical approach over the all-way bi-partition approach is about 77%. This experiment supports the claim we made in Section 3.

Cong and Lim reported that the multi-pass *gain-based* iterative all-way bi-partition is the best among all four

| circuit | multi-pass exhaus | | gain-based [7] | | %imp. |
|---------|-------------------|---------|----------------|---------|--------|
| | net-cut | time(s) | net-cut | time(s) | |
| fract | 64 | 23.5 | 75 | 20.8 | 14.7% |
| struct | 138 | 614.7 | 133 | 627.6 | -3.8% |
| p1 | 145 | 411.7 | 146 | 448.3 | 0.7% |
| p2 | 461 | 1816 | 455 | 1979 | -1.3 % |
| biomed | 305 | 1995 | 351 | 1833 | 13.1% |
| in1 | 210 | 2052 | 203 | 1259 | -3.4% |
| in2 | 979 | 5006 | 1001 | 4969 | 2.2% |
| in3 | 2224 | 3625 | 2210 | 5512 | -0.6% |
| avqs | 532 | 4485 | 585 | 21375 | 9.1% |
| avql | 539 | 3796 | 532 | 24315 | -1.3% |
| avg. | | | | | 2.9% |

TABLE II
MULTI-PASS EXHAUSTIVE ALL-WAY ALGORITHM VS. GAIN-BASED ALL-WAY ALGORITHM (CONCLUSION: THE TWO ALL-WAY APPROACHES ARE COMPARABLE IN QUALITY).

approaches described above [7]. However, they use partition results from a hierarchical algorithm as the input of the all-way bi-partition algorithm. When using random partitions as the input, experiments show that the gain-based all-way approach is actually worse than the exhaustive all-way approach. Table II shows this fact. The column of *multi-pass exhaustive* shows the results from a multi-pass exhaustive all-way algorithm. The column of *gain-based* shows the results from a multi-pass gain-based all-way algorithm which is implemented according to [7]. Table II shows that the multi-pass exhaustive algorithm performs slightly better than the gain-based algorithm.

In Table III, we compare the best results among all-way bi-partition algorithms to the hierarchical algorithm. The column of *% improv* shows the percentage improvement of the hierarchical algorithm vs. the best of the all-way algorithms. The column of *speedup* shows the speedup of the hierarchical algorithm vs. the best of the all-way algorithms. the hierarchical algorithm on average performs 7.1% better in quality and is 144 times faster than the all-way algorithm. This result suggests that given long enough time, the multi-pass all-way algorithm could get similar results as the hierarchical algorithm. Considering both quality and runtime, the hierarchical approach is better than the all-way bi-partition approach when using a random input partition.

The hierarchical approach is good to start with. However, the drawback of the hierarchical approach is that it is not effective to be used to improve an existing partition. The all-way approach can be easily used to improve an existing partition. In [7], Cong and Lim claimed that the gain-based all-way approach produces the best results when used after the hierarchical approach. Here we conduct an experiment to compare the exhaustive and the gain-based all-way approach used after the hierarchical

| circuit | best of allway | | hierarchical | | hie. vs. all-way | |
|---------|----------------|---------|--------------|---------|------------------|-------|
| | netcut | time(s) | netcut | time(s) | %imp. | spdup |
| fract | 64 | 23.5 | 56 | 1.21 | 12.5% | 19.4 |
| struct | 133 | 627.6 | 128 | 5.74 | 3.8% | 109 |
| p1 | 145 | 411.7 | 150 | 5.17 | -3.4% | 79.6 |
| p2 | 455 | 1979 | 450 | 12.5 | 1.1% | 158 |
| biomed | 305 | 1995 | 213 | 13.18 | 30.2% | 151 |
| in1 | 203 | 1249 | 175 | 9.68 | 13.8% | 129 |
| in2 | 979 | 5006 | 898 | 44.9 | 8.3% | 111 |
| in3 | 2210 | 5512 | 2070 | 80.1 | 6.3% | 68.8 |
| avqs | 532 | 4485 | 532 | 42.6 | 0% | 105 |
| avql | 532 | 24315 | 543 | 47.8 | -2.1% | 509 |
| avg. | | | | | 7.1% | 144 |

TABLE III
COMPARING ALL-WAY ALGORITHM VS. THE HIERARCHICAL ALGORITHM (CONCLUSION: THE HIERARCHICAL APPROACH IS MORE EFFECTIVE THAN THE ALL-WAY APPROACH).

approach. We take the partition results produced by the hierarchical approach (Table. III) as the input partition for our all-way approach. Table. IV shows the results of comparison. The results show that both algorithms can not improve over the hierarchical partitioning results. Thus if the we use a very good bi-partitioning tool like hMetis in the hierarchical scheme, the all-way approach is not useful even used after the hierarchical run.

V. CONCLUSION

In this paper, we introduced the concept of embedding graph to theoretically analyze different k -way partition algorithms. We showed that the cut cost upper bound for the hierarchical bi-partition approach is $O(\delta \log k)$ of the optimal result and the cut cost upper bound for the all-way bi-partition approach is $O(\delta k)$ of the optimal result assuming that we are using a δ -approximation bi-partition heuristic (defined in Section 3). These two upper bounds suggests that the hierarchical approach is a better way to solve the k -way partition problem. Experimental results strongly support this claim. When the target graph is large, the hierarchical approach can have a cut cost which is about 77% better than the single-pass all-way bi-partition approach. Experimental results also show that the all-way algorithm can improve given long enough time. However, the hierarchical approach is still **clearly the winner in quality and runtime**. Specifically, the hierarchical approach produces 7.1% better results and is 144 times faster than the all-way approach.

REFERENCES

- [1] C. J. Alpert, J. H. Huang, and A. B. Kahng. "Multilevel Circuit Partitioning". In *Design Automation*

| circuit | exhaustive after hie. | | gain-based after hie. | |
|---------|-----------------------|--------------|-----------------------|--------------|
| | netcut | %imp.vs.hie. | netcut | %imp.vs.hie. |
| fract | 54 | 3.6% | 54 | 3.6% |
| struct | 127 | 0.8% | 128 | 0% |
| p1 | 143 | 4.7% | 143 | 4.7% |
| p2 | 433 | 3.8% | 447 | 0.7% |
| biomed | 213 | 0% | 213 | 0% |
| in1 | 175 | 0% | 175 | 0% |
| in2 | 898 | 0% | 898 | 0% |
| in3 | 2070 | 0% | 2070 | 0% |
| avqs | 532 | 0% | 532 | 0% |
| avql | 543 | 0% | 532 | 2.0% |
| avg. | | 1.3% | | 1.1% |

TABLE IV

MULTI-PASS EXHAUSTIVE ALL-WAY ALGORITHM VS. GAIN-BASED ALL-WAY ALGORITHM WHEN USED AFTER THE HIERARCHICAL ALGORITHM (CONCLUSION: IT IS VERY HARD TO FURTHER IMPROVE THE HIERARCHICAL RESULTS USING THE ALL-WAY ALGORITHM.

- Conference*, pages 530–533. IEEE/ACM, 1997.
- [2] C. J. Alpert and A. B. Kahng. “Geometric Embeddings for Faster and Better Multi-Way Netlist Partitioning”. In *Design Automation Conference*, pages 743–748. IEEE/ACM, 1983.
- [3] C. J. Alpert and A. B. Kahng. “Recent Directions in Netlist Partitioning”. *Integration, the VLSI Journal*, 19:1–81, 1995.
- [4] C. K. Cheng and Y. C. A. Wei. “An improved two-way partitioning algorithm with stable performance”. *IEEE Transactions on Computer Aided Design*, 10(12):1502–1511, 1991.
- [5] J. Cong, W. Labio, and N. Shivakumar. “Multi-way VLSI circuit partitioning based on dual net representation”. In *International Conference on Computer-Aided Design*, pages 56–62, November 1994.
- [6] J. Cong, H. P. Li, S. K. Lim, T. Shibuya, and D. Xu. “Large Scale Circuit Partitioning with Loose/Stable Net Removal and Signal Flow Based Clustering”. In *International Conference on Computer-Aided Design*, pages 441–446. IEEE, 1997.
- [7] J. Cong and S. K. Lim. “Multiway Partitioning with Pairwise Movement”. In *International Conference on Computer-Aided Design*, pages 512–516, 1998.
- [8] D. Dutt and W. Deng. “VLSI Circuit Partitioning by Cluster-Removal Using Iterative Improvement Techniques”. In *International Conference on Computer-Aided Design*, pages 194–200. IEEE, 1996.
- [9] C. M. Fiduccia and R. M. Mattheyses. “A Linear Time Heuristic for Improving Network Partitions”. In *Design Automation Conference*, pages 175–181, 1982.
- [10] L. Hagen and A. B. Kahng. “Fast Spectral Methods for Ratio Cut Partitioning and Clustering”. In *International Conference on Computer-Aided Design*. IEEE, 1991.
- [11] S. Hauck and G. Boriello. “An Evaluation of Bipartitioning Techniques”. In *Chapel Hill Conference on Advanced Research in VLSI*, 1995.
- [12] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. “Multilevel Hypergraph Partitioning: Application in VLSI Domain”. In *Design Automation Conference*, pages 526–529. IEEE/ACM, 1997.
- [13] G. Karypis and V. Kumar. “Multilevel k-way Hypergraph Partitioning”. In *Design Automation Conference*, pages 343–348, 1999.
- [14] B.W. Kernighan and S. Lin. “An Efficient Heuristic Procedure for Partitioning Graphs”. *Bell System Technical Journal*, 49:291–307, February 1970.
- [15] M. Kiwi, D. Spielman, and S. H. Teng. “Min-Max-Boundary Domain Decomposition”. In *Annual International Computing and Combinatorics Conference*, 1998.
- [16] L. A. Sanchis. “Multi-Way Network Partitioning”. *IEEE Transactions on Computers*, 38(1):62–81, Jan. 1989.
- [17] H. D. Simon and S. H. Teng. “How Good Is Recursive Bisection?”. In *SIAM Journal of Scientific Computing*, 1996.
- [18] P. R. Suaris and G. Kedem. “Quadrisection: A New Approach to Standard Cell Layout”. In *Design Automation Conference*, pages 474–477. IEEE/ACM, 1987.
- [19] P. R. Suaris and G. Kedem. “Standard Cell Placement by Quadrisection”. In *International Conference on Computer-Aided Design*, pages 612–615. IEEE/ACM, 1987.
- [20] N. S. Woo and J. Kim. “An Efficient Method of Partitioning Circuits for Multiple-FPGA Implementation”. In *Design Automation Conference*, pages 202–207. IEEE/ACM, 1993.