A General Framework for Vertex Orderings, With Applications to Netlist Clustering*

C. J. Alpert and A. B. Kahng

UCLA Computer Science Department, Los Angeles, CA 90024-1596

Abstract

We present a general framework for the construction of *vertex orderings* for netlist clustering. Our WINDOW algorithm constructs an ordering by iteratively adding the vertex with highest *attraction* to the existing ordering. Variant choices for the attraction function allow our framework to subsume many graph traversals and clustering objectives from the literature. The DP-RP method of [3] is then applied to optimally split the ordering into a k-way clustering. Our approach is adaptable to user-specified cluster size constraints. Experimental results for clustering and multi-way partitioning are encouraging.

1 Introduction

A netlist hypergraph $H(V, E)$ consists of a set of modules (vertices) $V = \{v_1, v_2, \ldots, v_n\}$ and a set of nets (hyperedges) $E = \{e_1, e_2, \ldots, e_m\}$. A *cluster* C_i is a nonempty subset of V, and a *k-way clustering* P^k is a set of k clusters such that every $v_i \in V$ belongs to exactly one cluster in P^k. We study the following problem:

The k-Way Clustering Problem: Given $H(V, E)$, a value $2 \le k \le n$, and cluster size bounds L and U, construct $P^k = \{C_1, C_2, \ldots, C_k\}$ with $L \le |C_i| \le U$, $1 \le i \le k$, that optimizes a given objective function $f(P^k)$.

We often refer to P^k as a *clustering* when k is large, e.g., $k = \Theta(n)$, and as a *partitioning* when k is small, e.g., $k \le 10$. Where applicable, we let $w(C_i)$ denote the *cost* or weight of having C_i in the clustering, and express f in terms of w. Our work is motivated by two observations:

- Clustering reduces the problem size and can improve the performance of partitioning and placement heuristics [4, 15]. However, alternative clustering metrics must still be explored for such "meta-objectives".

- We need fast, high-quality clustering constructions that adapt to a variety of alternate objectives.

*Partial support for this work was provided by a Department of Defense Graduate Fellowship, and by NSF MIP-9257982 and MIP-9223740. The authors thank Dr. Lars Hagen for code and experimental data.

In this paper, we seek vertex orderings that capture the clustering structure of a netlist hypergraph, such that the vertices in any contiguous subset of the ordering form a "good" cluster. For this purpose, vertex orderings induced by traversals such as depth-first or breadth-first search are insufficient: (i) a DFS ordering can wander, rather than remain in a dense region; and (ii) a BFS ordering will visit all neighbors of a given vertex but can then jump to an entirely different region of the topology. Our main contribution is a new framework for traversing a graph to induce a vertex ordering: vertices are added into the ordering based on their *attraction* to the previous history of the ordering. This paradigm is fast and flexible – we exhibit attraction functions that can capture a variety of ordering constructions. The user can also set parameters to construct an ordering best suited to particular applications.

1.1 Clustering Methods and Metrics

Many clustering approaches have been proposed in the literature, e.g., [2, 4, 6, 9, 15]. These methods typically address meta-objectives such as the utility of the clustering within two-phase Fiduccia-Mattheyses [8] (FM) bisection or within annealing placement. At the same time, explicit clustering objectives such as the following have been proposed.

- The **DS** objective [7] is:

$$maximize \quad f(P^k) = \frac{1}{n}\sum_{i=1}^{k} w(C_i) \quad where$$

$$w(C_i) = |C_i| \cdot \frac{degree(C_i)}{separation(C_i)}$$

  Here, $degree(C_i)$ is the average number of nets incident to each module of the cluster that have at least two pins in the cluster; $separation(C_i)$ is the average length of a shortest path between two modules in C_i ($= \infty$ if the cluster is disconnected). This requires $O(n^3)$ time to evaluate, making DS more useful for comparison, rather than optimization, of clustering solutions.

- What we call the **Absorption** metric [15] counts the number of nets "absorbed" by the clusters:

$$maximize \quad f(P^k) = \sum_{i=1}^{k} w(C_i) \quad where$$

$$w(C_i) = \sum_{\{e \in E \mid e \cap C_i \neq \emptyset\}} \frac{|e \cap C_i| - 1}{|e| - 1}$$

i.e., net e incident to cluster C_i adds absorption $(p-1) \cdot \frac{1}{|e|-1}$ to the cluster, where p is the number of pins of e in the cluster.

- **Scaled Cost** [5] is a k-way generalization of the ratio cut objective:

$$minimize \quad f(P^k) = \frac{1}{n(k-1)} \sum_{i=1}^{k} w(C_i) \quad where$$

$$w(C_i) = \frac{|\{e \mid \exists u, v \in e, u \in C_i, v \notin C_i\}|}{|C_i|}$$

i.e., $w(C_i)$ is the "outdegree" of a cluster, divided by the cluster size.

1.2 Vertex Orderings

Given vertices $V = \{v_1, v_2, \ldots, v_n\}$, a *vertex ordering* $v_{\pi_1}, v_{\pi_2}, \ldots, v_{\pi_n}$ is defined by a bijection $\pi : [1 \ldots n] \rightarrow [1 \ldots n]$. Vertex v_i is the j^{th} vertex in the ordering if $\pi(j) = i$, so that v_{π_1} is the first vertex in the ordering, v_{π_2} is the second vertex, etc.

The Vertex Ordering Problem: Given $H(V, E)$, construct a vertex ordering $v_{\pi_1}, v_{\pi_2}, \ldots, v_{\pi_n}$ to optimize some objective.

Intuitively, contiguous subsets of the ordering should form "good clusters". Previous work has seen such contexts as ordering of one-dimensional logic arrays, graph partitioning, and sparse matrix computation. For example, Cuthill and McKee [13] proposed a BFS variant which breaks ties in favor of the vertex with smallest degree. King [13] proposed a *min-perimeter* approach which, when we view the set of ordered vertices as a single cluster, iteratively adds the vertex that minimizes the "perimeter" of the resulting cluster. Alternatively, adding the vertex with the most connections to the current cluster yields a *max-adjacency* approach; cf. Nagamochi and Ibaraki [12]. Other constructions been given for VLSI layout. Hall [11] showed that the second eigenvector of the netlist discrete Laplacian yields a minimum squared-wirelength ordering; [10] used this ordering in ratio cut partitioning. Riess et al. [14] used an analytical conjugate gradient method to construct orderings according to a linear wirelength objective. In [3], we induced (one-dimensional) orderings via spacefilling curves over multi-dimensional spectral netlist embeddings.

In the next section, we describe how an iterative graph traversal can encompass different orderings by varying an *attraction* function. Section 3 then proposes our WINDOW algorithm, as well as parameters that allow clusterings with user-specified attributes. Section 4 explains how WINDOW orderings can be split to yield a k-way clustering, and we conclude with experimental results.

2 The Attraction Function

Our general framework is as follows. We say that vertex v_j has been *ordered* if $\pi(index) = j$ for some *index*; v_j is otherwise *unordered.* In the following, we generally use v_j to indicate an ordered vertex, v_i for an unordered vertex, and v_{i^*} for the "best" unordered vertex. We also use $Nets(i) = \{e \in E \mid v_i \in e\}$ to denote the set of nets incident to v_i, and $Adj(i) = \{v_j \in e \cap S \mid e \in Nets(i)\}$ to denote the set of ordered neighbors of v_i. Let $S = \{v_j \in V \mid v_j \text{ is ordered}\}$, and for each unordered v_i let $Attract(i)$ be the *attraction* from v_i to S.

1. **Initialize:** Choose a vertex v_{i^*} and set $\pi(1) = i^*$. Set *index*, the current size of S, to 1. For each $v_i \in V - S$, compute $Attract(i)$.
2. **Best Vertex:** If $V - S \neq \emptyset$ choose $v_{i^*} \in V - S$ with optimal $Attract(i^*)$, else exit.
3. **Update:** Increment *index* and set $\pi(index) = i^*$. Update $Attract(i)$ for each $v_i \in V - S$ and go to Step 2.

Many traditional vertex orderings are captured by our framework.

- **DFS Ordering.** The attraction for v_i is:

$$Attract(i) = \max\{j \mid v_{\pi_j} \in Adj(i)\}$$

i.e., $Attract(i)$ is the index of v_i's most recently ordered neighbor. If $Adj(i) = \emptyset$, then $Attract(i) = 0$. The "best" vertex v_{i^*} will be adjacent to the most recently ordered vertex that has an unordered neighbor. Fig. 1(a) shows a snapshot of *Attract* values during construction of a DFS ordering, given that five vertices have already been ordered.

- **BFS Ordering.** The attraction for v_i is:

$$Attract(i) = \min\{j \mid v_{\pi_j} \in Adj(i)\}$$

If $Adj(i) = \emptyset$, then $Attract(i) = \infty$. The best vertex v_{i^*} has minimum $Attract(i^*)$ (see Fig. 1(b)).

- **Max-Adjacency Ordering.** The attraction for v_i is:

$$Attract(i) = |\{e \in Nets(i) \mid e \cap S \neq \emptyset\}|$$

The best vertex v_{i^*} has the most hyperedges incident to vertices in S (see Fig. 1 (c)).

- **Absorption Metric.** The attraction for v_i is:

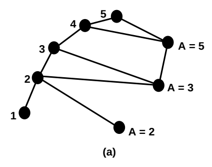
$$Attract(i) = \sum_{\{e \in Nets(i) \mid e \cap S \neq \emptyset\}} \frac{1}{|e| - 1}$$

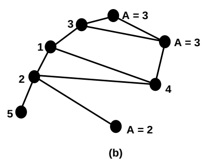
For each e incident to v_i and S, absorption $\frac{1}{|e|-1}$ is gained by adding v_i to S. Thus, v_{i^*} that maximizes $Attract(i^*)$ will give the greatest increase in the absorption of S.

- **Scaled Cost Metric.** An attraction function that is based directly on Scaled Cost may not be effective, since the perimeter of a cluster does not measure how close the cut nets are to becoming uncut. Thus, we use:

$$Attract(i) = \sum_{e \in Adj(i)} \frac{|S \cap e|}{|e| - 1}$$

A net $e \in Nets(i)$ exerts attraction on v_i proportional to the number of its pins in S. As more vertices of a given net become ordered, the unordered vertices incident to that net feel stronger attraction to the ordering. By contrast, a net e incident to v_i and S would exert the same Absorption attaction on v_i regardless of how many pins of e are in S.





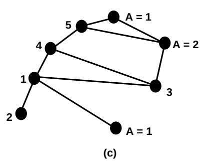


Figure 1: Snapshots of feasible orderings: (a) DFS, (b) BFS, and (c) Max-Adjacency. Ordered vertices are labeled by indices in the ordering; unordered vertices are labeled by attraction value A. With (a) and (c), v_{i^*} maximizes A; with (b), v_{i^*} minimizes A.

Naive updates of $Attract(i)$ in Step 3 may result in an $O(n^2)$ ordering construction. However, for many attraction functions $Attract(i)$ increases (decreases) monotonically throughout the ordering process. Thus, our implementation uses a Fibonacci heap to store each $v_i \in V - S$ with $Attract(i)$ as the corresponding key: the vertex with maximum (minimum) key is iteratively extracted from the heap, and keys for the other vertices are updated via an increase-key (decrease-key) operation. This results in an amortized $O(n \log n)$ time implementation.

Notice that the attraction functions listed above treat all of S as the "current cluster". However, if the ordering is to be subsequently split into a k-way clustering, vertices ordered earlier will probably not end up in the same clusters as vertices ordered later. This suggests that only the ordered vertices which can potentially belong to v_i's cluster - i.e., the more recently ordered vertices - should exert attraction on v_i. Based on this idea, our new WINDOW algorithm is developed as follows.

3 The WINDOW Construction

We define the current *window* of size W to consist of the W most recently ordered vertices; only vertices in the window exert full attraction on unordered vertices. Notice that a user might set $W = \frac{n}{k}$ in seeking a k-way clustering over n vertices. However, some reflection reveals that this simple approach cannot adequately deal with possible cluster size bounds L and U. For example, if we have $W = \frac{n}{k} = 5$ and $L = 1$, this framework would yield the same ordering whether $U = 10$ or $U = 1000$; in the latter case, as many as 994 ordered vertices might end up sharing a cluster with v_{i^*} after having no influence on the choice of this vertex. Hence, we use a second parameter T to define the *tail* of the window; attraction exerted by a vertex in the tail is proportional to distance from the end of the window. Figure 2 depicts the attraction exerted by the first 100 ordered vertices for different choices of W and T. Figure 3 integrates the window and tail concepts into a description of our WINDOW construction.

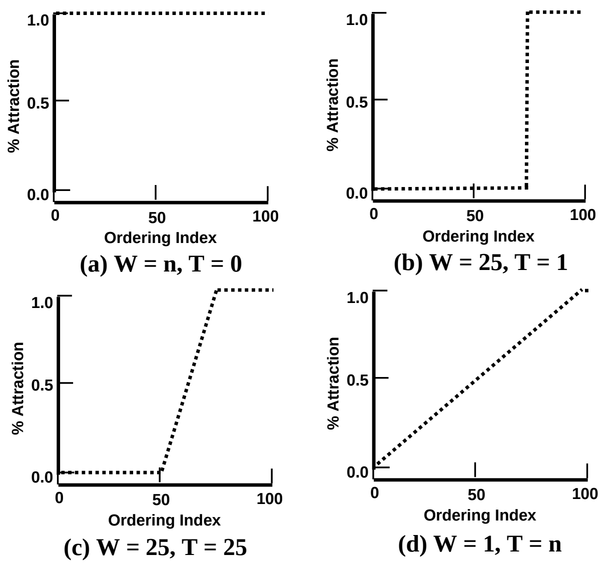


Figure 2: Attraction exerted by the first 100 ordered vertices for varying W and T values.

```
The WINDOW Algorithm
Input:   Netlist $H(V,E)$
         $W \equiv$ window size
         $T \equiv$ tail of the window
         Objective function $Attract$
Output:  Vertex ordering $v_{\pi_1}, v_{\pi_2}, \ldots, v_{\pi_n}$
1. Choose a vertex $v_{i^*} \in V$ and set $v_{\pi_1} = v_{i^*}$
   for each unordered $v_i \in V$, do compute $Attract(i)$.
2. for $index = 2$ to $n$ do
3.     Choose unordered $v_{i^*}$ with optimal $Attract(i^*)$
4.     Set $v_{\pi_{index}}$ to $v_{i^*}$
5.     for each unordered $v_i$, update $Attract(i)$ such that
         if $index - W + 1 \le j \le index$, then
            vertex $v_{\pi_j}$ has full attraction on $v_i$
         if $index - W - T + 1 \le j \le index - W$, then
            vertex $v_{\pi_j}$ has attraction $\frac{(T+W+j-index)}{T}$ on $v_i$
6. return $v_{\pi_1}, v_{\pi_2}, \ldots, v_{\pi_n}$
```

Figure 3: The WINDOW Algorithm

We observe that if $T = 1$ (i.e., the window has no tail) and W is a constant, then WINDOW can typically be implemented to run in linear time. Let N denote the set of unordered neighboring vertices of the window. Since netlist modules have bounded degree, $|N|$ is bounded by a constant proportional to W. The Step 5 updating can be done by adding unordered neighbors of the chosen v_{i^*} to N and then updating $Attract(i)$ for each $v_i \in N$; minor additional bookkeeping reflects the shift of the new window by one position. For $T > 1$, Step 5 takes time proportional to T, and the complexity becomes $O(nT|N|)$.

4 Splitting Orderings into Clusterings

In [3], we presented the "DP-RP" algorithm for constructing a multi-way partitioning. DP-RP accepts a vertex ordering as input and returns a *restricted partitioning*, i.e., a k-way partitioning with each cluster being a contiguous subset of the ordering. Dynamic programming is used to find the optimal set of $k-1$ splits of the ordering that induce the k-way partitioning; this is possible for any clustering objective that is a *monotone* function of an intercluster cost metric (e.g., both Absorption and Scaled Cost are expressible as monotone functions in w). Although the complexity of DP-RP depends on the objective function, $O(nU + kn(U-L))$ implementations have been given for Scaled Cost [3] and Absorption [1].

5 Experimental Results

5.1 Clustering Comparisons

Table 1 compares DP-RP clusterings derived from WINDOW vertex orderings with the MBC [4], RW-ST [9], and AGG [2] clusterings, in terms of the Scaled Cost, Absorption, DS and two-phase FM min-cut bisection measures. The same number of clusters is used as in the experiments of [9, 2]. For AGG, we report the best Scaled Cost and Absorption values over the ten AGG clusterings available for each test case; we also report the best min-cut over 200 FM runs (20 for each clustering). Because computing the DS metric requires $O(n^3)$ time, we report only the DS value for the AGG clustering with lowest Scaled Cost. FM min-cuts for the other algorithms are the best observed over 20 runs and are quoted from [2] with the exception of the Test05, for which FM min-cuts were regenerated due to a faulty area file used in the original experiments.

The WINDOW clusterings were generated using cluster size constraints $L = 1$ and $U = 20$, and with $W = \lceil \frac{n}{k} \rceil$ and $T = U - W$. A random *pseudo-peripheral* vertex was used to begin the ordering.[1] (Separately, we have found that these parameters are not optimal, and that in particular the tradeoffs between W and T remain unclear [1].)

Notice that because each ordering is derived using the appropriate attraction function, the WINDOW results for Scaled Cost and Absorption correspond to different clusterings.[2] As one would expect, when WINDOW optimizes one objective, the clustering usually worsens with respect to other objectives. For example, with the Primary1-SC test case, minimizing Scaled Cost (173.1) leads to Absorption = 621.9, while maximizing Absorption (687.6) leads to Scaled Cost = 234.8.

For Scaled Cost, Absorption and DS, WINDOW clusterings averaged 34.2%, 13.2% and 8.3% respective improvement versus the closest other results. For two-phase FM, WINDOW results were comparable to MBC and RW-ST, but inferior to AGG. Improvements may be possible using alternative attraction functions and less restrictive cluster size bounds. CPU times for our methodology on a Sun Sparc-10 were 9.7, 36 and 63 second to generate orderings for Primary2, Biomed and Industry2 respectively; the additional times for DP-RP to construct the clusterings were 106, 385, and 1322 seconds for the same three instances.

Case	Alg	SC	Absrp	DS	FM
Pr1-SC	WINDOW	173.1	687.6	1.471	48
833	RW-ST	287.9	629.9	1.325	47
(191)	AGG	277.9	437.0	0.879	49
	MBC	254.0	309.3	1.258	48
Pr2-SC	WINDOW	57.69	2257	1.539	186
3014	RW-ST	82.81	2013	1.566	165
(702)	AGG	89.73	1227	1.048	146
	MBC	82.44	736.4	1.238	187
Test02	WINDOW	97.02	1327	1.662	42
1663	RW-ST	150.7	1123	1.593	42
(445)	AGG	164.7	706.2	0.657	42
	MBC	137.4	407.0	1.231	42
Test03	WINDOW	91.50	1247	1.736	53
1607	RW-ST	156.2	1101	1.566	71
(327)	AGG	153.9	678.4	1.204	50
	MBC	140.7	379.5	1.185	59
Test04	WINDOW	100.2	1303	2.014	20
1515	RW-ST	151.8	1181	1.879	14
(317)	AGG	193.3	833.9	1.135	12
	MBC	160.3	415.5	1.297	20
Test05	WINDOW	55.28	2279	1.831	36
2595	RW-ST	88.52	2051	1.689	28
(424)	AGG	103.0	1527	1.262	32
	MBC	90.08	680.7	1.275	37
Test06	WINDOW	106.9	1274	1.516	73
1752	RW-ST	178.7	979.2	1.367	82
(476)	AGG	163.2	359.0	1.183	63
	MBC	142.4	315.3	1.331	83
19ks	WINDOW	47.51	2556	1.883	127
2844	RW-ST	81.08	2395	1.578	146
(737)	AGG	86.50	1485	1.022	124
	MBC	75.50	719.9	1.166	156
bm1	WINDOW	137.8	692.5	1.278	62
882	RW-ST	258.5	637.9	1.221	58
(216)	AGG	266.0	426.6	0.813	48
	MBC	340.5	199.1	1.189	54

Table 1: Comparison between WINDOW and four other clustering algorithms. The numbers below each test case indicate n and k. MBC, RW-ST, and AGG clusterings were obtained from [9] and [2].

5.2 Vertex Ordering Comparisons

We also compared WINDOW orderings to five other vertex ordering constructions: King, Cuthill-McKee (CM) [13], Max-Adjacency (MA) [12], EIG1 [10] and SFC [3]. We ran DP-RP on the vertex ordering constructed by each algorithm for each test case, again with $L = 1$, $U = 20$ and k as specified in Table 1. Tables 2 and 3 respectively provide the Scaled Cost and Absorption values for the clusterings generated by DP-RP. SFC results are the best results obtained from ten SFC orderings.

[1] The *eccentricity* of a vertex v is the distance of the vertex u furthest from v. A *pseudo-peripheral vertex* v has the property that if the distance from u to v is also the eccentricity of v, then the eccentricity of u is no larger than the eccentricity of v.

[2] It is not clear which attraction functions are best for DS and two-phase FM. Furthermore, DP-RP is very inefficient when applied to DS, and cannot be applied at all to two-phase FM. Thus, we measured DS and FM cuts in terms of the WINDOW clusterings that optimized Scaled Cost.

Case	Algorithm					
	King	CM	MA	EIG1	SFC	WIN
Pr1-SC	209.6	226.7	176.1	244.1	204.7	173.1
Pr2-SC	68.97	73.24	61.08	78.82	68.10	57.69
Test02	118.1	129.6	100.3	137.4	131.4	97.02
Test03	119.6	130.1	97.28	132.1	126.5	91.50
Test04	132.3	143.9	109.2	155.4	147.7	100.2
Test05	73.27	83.97	60.42	85.08	76.06	55.28
Test06	121.8	127.5	107.0	131.5	142.8	106.9
19ks	63.12	69.79	50.18	73.58	66.37	47.51
bm1	168.3	185.9	132.8	184.8	146.2	137.8
Biomed	26.95	28.99	21.51	32.53	22.98	20.91
Indstr2	14.28	15.39	11.73	16.95	13.25	11.35

Table 2: Scaled Cost values for clusterings derived from six ordering constructions.

Case	Algorithm					
	King	CM	MA	EIG1	SFC	WIN
Pr1-SC	489.0	259.6	534.6	312.8	513.3	687.6
Pr2-SC	1209	598.6	1355	600.2	1114	2257
Test02	709.8	282.9	991.4	468.8	618.6	1327
Test03	649.3	300.6	951.7	525.5	640.3	1247
Test04	716.8	326.8	992.3	382.6	625.3	1303
Test05	1069	490.8	1562	891.9	1324	2279
Test06	546.5	247.6	765.3	264.9	349.1	1274
19ks	1389	636.2	2042	806.2	1573	2556
bm1	512.8	278.0	546.4	355.5	513.8	692.5
Biomed	2134	1427	3608	491.1	3403	5070
Indstr2	4392	1774	7551	1987	4630	10747

Table 3: Absorption values for clusterings derived from six ordering constructions.

For the first nine test cases, WINDOW obtains fairly consistent improvements for both metrics. For Scaled Cost, we observed 4.0% improvement over the closest other algorithm, Max-Adjacency. However, WINDOW subsumes Max-Adjacency when the proper attraction function and $W = n$ are used. Discounting Max-Adjacency, we observed 18.3% average reduction in Scaled Cost over the best combined results of the other algorithms. For Absorption, WINDOW obtained a 39.5% average improvement (increase) over Max-Adjacency, and 78.3% improvement over the combined results of the other four ordering constructions. For the larger test cases (Biomed: $n = 6514$, $k = 1303$; and Industry2: $n = 12637$, $k = 2527$), WINDOW-derived clusterings had lowest Scaled Cost and highest Absorption, with Max-Adjacency again being quite competitive. Our results seem to suggest that "global" EIG1 and SFC orderings cannot make the local decisions necessary for DP-RP to generate a good clustering when k is large.

In conclusion, we have developed a general framework for constructing vertex orderings, and explored its applications to netlist clustering. By setting an appropriate "attraction" function and window size, we obtained superior clusterings for a variety of clustering objectives in the literature. We leave open the question of finding improved attraction functions for meta-objectives that cannot be defined explicitly, such as two-phase FM enhancement.

References

[1] C. J. Alpert and A. B. Kahng, "A General Framework for Vertex Orderings, With Applications to Netlist Clustering," *UCLA tech. report #940018*, April 1994.

[2] C. J. Alpert and A. B. Kahng, "Geometric Embeddings for Faster and Better Multi-way Netlist Partitioning," *Proc. ACM/IEEE Design Automation Conf.* 1993, pp. 743-748.

[3] C. J. Alpert and A. B. Kahng, "Multi-way Partitioning Via Spacefilling Curves and Dynamic Programming," *Proc. ACM/IEEE Design Automation Conf.*, 1994, pp. 652-657.

[4] T. N. Bui, "Improving the Performance of the Kernighan-Lin and Simulated Annealing Graph Bisection Algorithms", in *Proc. ACM/IEEE Design Automation Conf.*, 1989, pp. 775-778.

[5] P. K. Chan, M. D. F. Schlag and J. Zien, "Spectral K-Way Ratio Cut Partitioning and Clustering", *Proc. Symp. on Integrated Systems*, Seattle, March 1993.

[6] J. Cong and M. Smith "A Parallel Bottom-up Clustering Algorithm with Applications to Circuit Partitioning in VLSI Design" *Proc. ACM/IEEE Design Automation Conf.* 1993, pp. 755-760.

[7] J. Cong, L. Hagen and A. B. Kahng, "Random Walks for Circuit Clustering", *Proc. 4th IEEE Intl. ASIC Conf.*, Rochester, September 1991, pp. 14.2.1 - 14.2.4.

[8] C.M Fiduccia and R.M. Mattheyses, "A Linear Time Heuristic for Improving Network Partitions", *Proc. ACM/IEEE Design Automation Conf.*, June 1982, pp. 175-181.

[9] L. Hagen and A. B. Kahng, "A New Approach to Effective Circuit Clustering", *Proc. IEEE Intl. Conf. on Computer-Aided Design*, Santa Clara, Nov. 1992, pp. 422-427.

[10] L. Hagen and A. B. Kahng, "New Spectral Methods for Ratio Cut Partitioning and Clustering", *IEEE Trans. on CAD* 11(9), Sept. 1992, pp. 1074-1085.

[11] K.M. Hall, "An r-dimensional Quadratic Placement Algorithm", *Manag. Sci.*, 17(1970), pp.219-229.

[12] H. Nagamochi and T. Ibaraki, "Computing Edge-Connectivity in Multigraphs and Capacitated Graphs", *Siam J. of Disc. Math.* 5(1), Feb. 1992, pp. 54-66.

[13] S. Pissanetsky, *Sparse Matrix Technology*, Academic Press Inc., 1984.

[14] B. M. Riess, K. Doll, and F. M. Johannes, "Partitioning Very Large Circuits Using Analytical Placement Techniques", *Proc. ACM/IEEE Design Automation Conf.*, 1994, pp. 646-651.

[15] W. Sun and C. Sechen, "Efficient and Effective Placements for Very Large Circuits" *Proc. IEEE Intl. Conf. on Computer-Aided Design*, Santa Clara, Nov. 1993, pp. 170-177.