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

C. J. Alpert and A. B. Kahng

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

Abstract

Vertex orderings have been successfully applied to problems in netlist clustering, for system partitioning and layout [2] [20]. We present a vertex ordering construction that encompasses most reasonable graph traversals. Two parameters – an *attraction* function and a *window* – provide the means for achieving various graph traversals and addressing particular clustering requirements. We then use dynamic programming [2] to optimally split the vertex ordering into a multi-way clustering. Our approach outperforms several clustering methods in the literature in terms of three distinct clustering objectives [5] [8] [22]. The ordering construction, by itself, also outperforms existing graph ordering constructions [2] [13] [17] [19] for this application. Tuning our approach to "meta-objectives", particularly clustering for two-phase Fiduccia-Mattheyses bipartitioning [9], remains an open area of research.

Keywords: Circuit and graph clustering, vertex ordering, graph traversal, partitioning, breadth-first search, depth-first search, maximum adjacency, minimum perimeter.

1 Introduction

Clustering of netlist hypergraphs has been used to reduce the complexity of VLSI CAD problem instances, particularly for system partitioning and layout. For example, clustering improves Fiduccia-Mattheyses (FM) bisection, via the "two-phase" methodology [4] where FM is first run on the clustered netlist, and the result provides the starting point of a second FM run on the flattened netlist. Sun and Sechen [22] have shown that clustering significantly improves simulated annealing placement via a similar two-phase approach.[1]

*Partial support for this work was provided by a Department of Defense Graduate Fellowship, and by NSF MIP-9257982 and MIP-9223740. A preliminary version of this work appeared in *Proc. IEEE Intl. Conf. on Computer-Aided Design*, Nov. 1994, pp. 63-67.

[1]Such efforts confirm the intuition of Bui et al. [4] and Lengauer [16] that good local minima in the flattened instance are more easily reached from good local minima in the clustered instance.

In this paper, we seek vertex orderings that capture the clustering structure of a given netlist, in the sense that the vertices in any contiguous subset of the ordering form a "good" (connected, dense, etc.) cluster. Such orderings are generally useful for minimizing wirelength in a 1-dimensional placement, minimizing the bandwidth of a sparse symmetric matrix [19], or other standard applications.

Given a vertex ordering, it is known that dynamic programming can be used to optimally split the ordering into a clustering [2]. However, vertex orderings induced by classic graph traversals such as depth-first search (DFS) or breadth-first search (BFS) are ill-suited to this purpose: (i) a DFS ordering may leave, rather than explore, a dense region; and (ii) a BFS ordering may visit all neighbors of a given vertex but then jump to an entirely different region of the topology.

We develop a new vertex ordering framework which encompasses many known graph traversals and can be fine-tuned to particular clustering requirements. Our ordering framework centers around the ideas of an *attraction* function and a *window*. Unordered vertices are iteratively added to the ordering based on their attraction to the previously ordered vertices. Various choices of attraction can capture DFS, BFS, max-adjacency [17], min-perimeter [18], and other well-studied graph traversals. Other choices can capture the intuition behind various VLSI clustering objectives such as *Scaled Cost* [5] and *Absorption* [22]. The window parameter allows the framework to adapt to possible cluster size constraints, by limiting attraction to only the most recently ordered vertices. While our focus is on circuit clustering, we believe that our construction directly applies to such problem classes as ordering of one-dimensional logic arrays, top-down system partitioning, module placement, and even sparse matrix computations.

We formalize the k-way clustering problem as follows. A system may be represented by its netlist hypergraph $H(V, E)$ consisting of a set of vertices (modules) $V = \{v_1, v_2, \ldots, v_n\}$ and a set of hyperedges (signal nets) $E = \{e_1, e_2, \ldots, e_m\}$. A signal net $e \in E$ is a subset of V with $|e| \geq 2$, and if $v_i \in e$, we say v_i is a *pin* of e. 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.

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

2 Clustering and Vertex Orderings

Clustering Methods and Objectives

Among the clustering approaches in the VLSI CAD literature, many have been proposed in the context of two-phase FM partitioning. The linear-time Matching Based Compaction (MBC) algorithm of Bui et al. [4] finds a random maximal matching in the netlist and "compacts" the matched pairs of modules into $\frac{n}{2}$ clusters. The random matching can be repeated if desired. Hagen and Kahng [12] suggested that a random-walk method (RW-ST) could find dense regions of the netlist, with "cycles" in the random walk forming the clusters. In another direction, Alpert and Kahng [3] and Chan et al. [5] apply geometric or hybrid geometric-topological clustering algorithms to multi-dimensional spectral embeddings of the netlist. Cong and Smith [7] present a bottom-up clustering method that recursively finds and collapses small, dense subgraphs in the netlist. These approaches all share a common "static" nature: it is not clear how to adapt them to particular clustering objectives, and with the exception of MBC, they are all somewhat complicated.

Two other works that address specific clustering objectives should also be noted. Sun and Sechen [22] construct clusterings for two-phase placement; their algorithm starts with modules assigned to k "bins" (clusters) and uses simulated annealing to reassign the modules among bins according to what we call the *Absorption* objective. For FPGA partitioning, Chou et al. [6] propose a local, iterative method that optimizes *Scaled Cost*, a multi-way generalization of the ratio-cut objective originally defined in [5]. In what follows, we will focus on these two objectives, which are defined as follows:[2]

- The **Absorption** objective [22] counts the number of nets "absorbed" by the clusters:

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

[2]In addition to the objectives that we discuss, Garbers et al. [11] have proposed a k-l connectivity criterion. Two vertices are k-l-connected if there are k edge-disjoint paths of length $\leq l$ between them, and the transitive closure of this relation defines the clustering. While there is some flexibility in that k and l are specified by the user, evaluating the objective is infeasible for $l > 2$ (and is known to be NP-hard for $l > 4$). We will also note experimental results for the **DS** objective [8], which is defined as:

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

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). Since DS requires $O(n^3)$ time to evaluate, it is most useful for comparison, rather than optimization, of clustering solutions.

where

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

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. Put another way: if the i^{th} net e_i has its pins distributed among c_i clusters, then $f(P^k) = \sum_{i=1}^{m} \frac{|e_i| - c_i}{|e_i| - 1}$.

- **Scaled Cost** [5] is defined as:

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

where the cost of a cluster is

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

i.e., $w(C_i)$ is the cluster "degree", divided by its size.

Vertex Orderings

Given the set of vertices $V = \{v_1, v_2, \ldots, v_n\}$, a *vertex ordering* $v_{\pi_1}, v_{\pi_2}, \ldots, v_{\pi_n}$ is represented by the 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. Generally, a vertex ordering objective tries to preserve the structure of the netlist: strongly connected modules should be near each other in the ordering, and a contiguous subset of the ordering should likely form a "good cluster" according to the above metrics. Vertex ordering objectives and methods have arisen in a variety of applications; the following provides a brief sampling.

For sparse matrix computations, rows of a symmetric $n \times n$ matrix are reordered to minimize the *bandwidth* or *profile* of the matrix, enabling efficient storage schemes (see [19] for a survey). Row reordering is also useful for reducing the *fill-in*, i.e., nonzeros added into the matrix when Gaussian elimination is performed. Since non-zeros in an $n \times n$ symmetric matrix can be viewed as edges in an n-vertex undirected graph, the row reordering in the matrix corresponds directly to vertex ordering in a graph:

- The Cuthill-McKee algorithm for minimizing bandwidth is a BFS variant in which ties are broken in favor of the vertex with smallest degree.

- The King algorithm is a *min-perimeter* approach: it builds a cluster by iteratively adding the vertex that minimizes the perimeter of the cluster. The sequence that vertices are added to the cluster determines the ordering.

For the construction of sparse graph spanners, Peleg and Schaffer [18] have applied a *max-perimeter* approach. In computing edge-connectivity of unweighted graphs, Nagamochi and Ibaraki [17] use a *max-adjacency* ordering: their algorithm iteratively adds the unordered vertex with the most edges incident to previously-ordered vertices.[3]

Finally, there are several vertex orderings that are implicit in heuristic global minimization of wirelength functions in one-dimensional module placements. Hall [14] showed that the second eigenvector of the discrete Laplacian of the netlist gives the optimal linear ordering of modules for minimum squared wirelength (subject to certain constraints); we call this the EIG1 ordering (after [13], who used this eigenvector for ratio-cut partitioning). Recently, Riess et al. [20] have used an analytical conjugate gradient method to construct module orderings according to the linear wirelength objective; their PARABOLI orderings improve ratio-cut partitionings by 50% over EIG1 orderings. Alpert and Kahng [2] have induced (one-dimensional) module orderings via spacefilling curves over multi-dimensional spectral netlist embeddings.

3 The Attraction Function

We say a vertex v_j has been *ordered* if $\pi(index) = j$ for some $index$; otherwise v_j is *unordered.* (Generally, v_j will indicate an ordered vertex, v_i an unordered vertex, and v_{i^*} the "best" unordered vertex.) Let $S = \{v_j \in V \mid \exists\ index,\ \pi(index) = j\}$ be the set of ordered vertices. We also let $Nets(i) = \{e \in E \mid v_i \in e\}$ denote the set of nets incident to v_i, and $Adj(i) = \{v_j \in e \mid e \in Nets(i), i \neq j\}$ denote the set of vertices that are adjacent to v_i. For each v_i, let $Attract(i)$ be the *attraction* from v_i to S. Our general framework is given in Figure 1.

The derived ordering π depends on the choice of both the first vertex and the attraction function. We now show how our framework can capture virtually any traditional vertex ordering construction.

[3] In VLSI CAD, the min-perimeter and max-adjacency orderings are at the heart of "direct" clustering methods used in early placement and partitioning approaches [16]. Similar orderings have been applied to the one-dimensional gate assignment problem (e.g., [10]).

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 $Attract(i^*)$, otherwise 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.

Figure 1: Vertex Ordering Framework.

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

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

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, say v_{π_j}, that has an unordered neighbor. In a depth-first traversal, v_{π_j} is the vertex to which we backtrack when the more recently ordered vertices $v_{\pi_{j+1}}, v_{\pi_{j+2}}, \ldots$ have no "unvisited" neighbors. Thus, v_i^*, an unvisited neighbor of v_{π_j}, is the next vertex visited by DFS. Figure 2(a) shows a snapshot of the *Attract* values during the construction of a DFS ordering, given that five vertices have already been ordered.

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

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

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

- **Max-Adjacency Ordering.** For general graphs, the authors of [17] define a traversal where the vertex v_i^* with the most edges incident to S is iteratively added to the ordering, i.e., v_i^* is *maximally adjacent* to S. This construction can be extended to hypergraphs by defining the attraction to be:

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

Vertex v_i^* belongs to the greatest number of nets incident to S (see Fig. 2 (c)).

- **Min-Perimeter (King) Ordering.** We want to minimize the number of unordered vertices incident to S. If $P = \{v_i \in V - S \mid \exists v_j \in S, v_j \in Adj(i)\}$ denotes the vertices that are currently in the perimeter of S, the attraction becomes:

$$Attract(i) = |\{Adj(i) \cup \{v_i\}\} \setminus \{S \cup P\}| - 1$$

$Attract(i)$ is the increase in the number of perimeter vertices that result by adding v_i to S.

- **The Absorption Objective.** If we regard S as a cluster, the best vertex v_{i^*} causes the new cluster $S \cup \{v_i^*\}$ to have maximum Absorption. Attraction is thus simply the *increase* in Absorption:

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

For each e incident to v_i and S, an Absorption of $\frac{1}{|e|-1}$ is gained by adding v_i to S.

- **Min-Cut/Scaled Cost Objective.** The Scaled Cost for the set S is the number of nets cut by S divided by the size of S. Since the size of S will increase by one regardless of which vertex is chosen, the Scaled Cost and min-cut objectives are equivalent.[4] The attraction function that exactly reflects the decrease in cut nets is given by

$$Attract(i) = |\{e \in Nets(i) \mid e \subseteq S \cup \{v_i\}\}| - |\{e \in Nets(i) \mid e \cap S = \emptyset\}|$$

The first term is the number of nets incident to v_i that will become uncut and the second term is the number of newly cut nets. However, this attraction function may not be effective in practice, since many ties will result. For example, say that e_1 and e_2 are two 10-pin nets incident to S, with $e_1 \cap S = 4$ and $e_2 \cap S = 8$. An unordered vertex incident to either net will not "uncut" the net, so a vertex incident to one of these nets will have attraction 0. However, a vertex incident to e_2 is preferred over a vertex incident to e_1 (or even to one with no connections to S), since e_2 is much closer to becoming uncut. We thus adopt the following heuristic attraction function:

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

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 e become ordered, the unordered vertices in e feel stronger attraction to

[4] Although these objectives are of course different, they are equivalent in terms of our framework and hence the same ordering is derived. However, when the DP-RP algorithm splits the ordering into a clustering, the two objectives are distinguishable and different clusterings will result.

S. (By contrast, according to the Absorption attraction function, a net e incident to v_i and S will exert the same attraction on v_i regardless of how many pins of e are in S.)

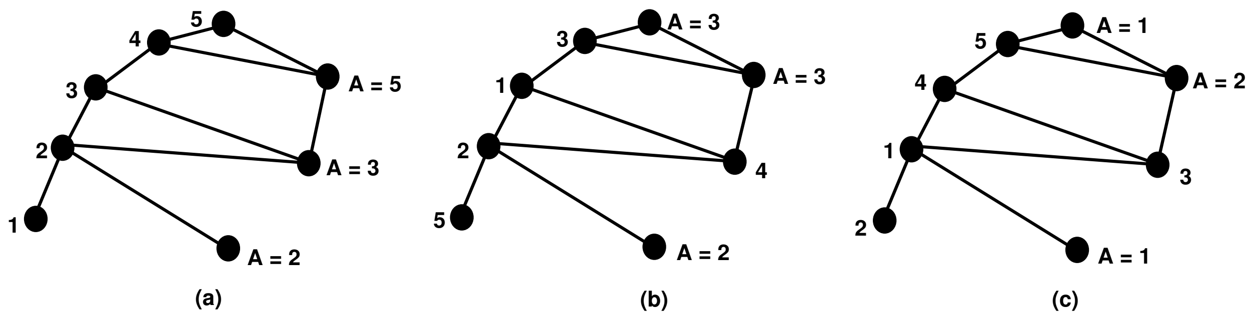

Figure 2: Snapshots for feasible orderings for (a) DFS, (b) BFS, and (c) Max-Adjacency. Ordered vertices are labelled by indices in the ordering; unordered vertices are labelled attraction value A. With (a) and (c), v_i^* maximizes A; with (b), v_i^* minimizes A.

Naive updates of *Attract*(i) in Step 3 of the ordering framework may result in an $O(n^2)$ 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. Assuming that the number of pins is $O(n)$ and that net sizes are bounded by a constant, this implementation requires $O(n \log n)$ amortized time.

4 The Window of Attraction

The above attraction functions treat all of S as the "current cluster". However, notice that if the ordering is subsequently split into a k-way clustering, earlier vertices in the ordering are unlikely to be in the same clusters as later vertices. Ideally, only the ordered vertices which can potentially belong to v_i^*'s cluster - i.e., the most recently ordered vertices - should exert attraction on v_i^*. To achieve this, we adopt the notion of a *window*: the current window of size W is the set of the W most recently ordered vertices, and only vertices in the window exert full attraction on unordered vertices.

In seeking a k-way clustering, one might, for example, set $W = \frac{n}{k}$, the average cluster size. This window size enables enables every vertex in the window to possibly share a cluster with the next vertex v_i^* added to the ordering. However, this simple approach does not adequately handle possible cluster size bounds L and U. For example, if we have $W = \frac{n}{k} = 5$, $L = 1$, and $U = 10$, up to five vertices may end up in the same cluster as v_i^* after having had no influence on the choice of this vertex. Furthermore, we would obtain the same ordering whether $U = 10$ or $U = 1000$; in the latter case, as many as 994 ordered vertices may be clustered with v_{i^*} after having had no influence on its choice. Hence, we use a second parameter T to define the size of the *tail* of the window. Vertices in the tail have some influence on the choice of v_i^*, but not to the same extent as vertices in the window. The attraction exerted by a vertex in the tail is proportional to its distance from the end of the window. Figure 3 depicts the attraction exerted by the first 100 ordered vertices for different choices of W and T.

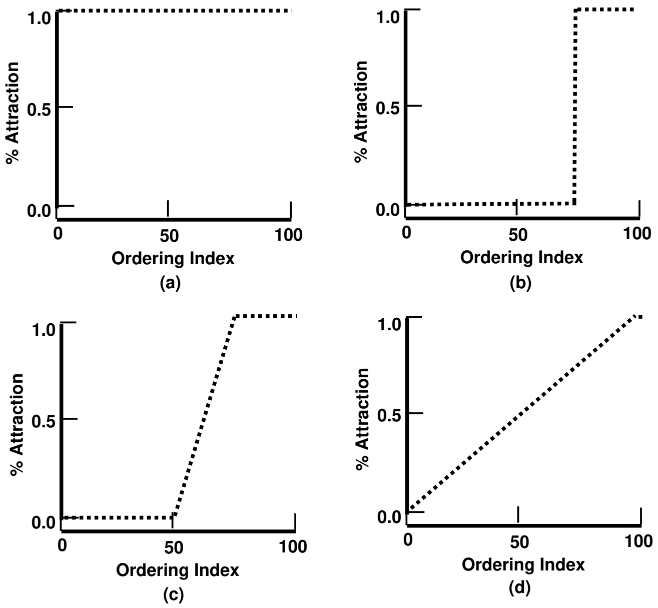

Figure 3: Attraction exerted by the first 100 ordered vertices for varying W and T values. (a) corresponds to the original framework of Section 3; (b) corresponds to a window of size 25 with no tail; (c) corresponds to a window of size 25 and a tail of size 25; and (d) corresponds to a "full-history" ordering approach where each subsequently ordered vertex exerts more attraction than previously ordered vertices ($W = 1$ and $T = n$).

Figure 3 integrates the concepts of a window and tail into our framework (called the WINDOW algorithm). If $T = 1$ and W is a constant, then WINDOW can typically be implemented in linear time. Let N denote the set of unordered neighboring vertices of the current window. Since netlist modules have bounded degree, $|N|$ is bounded by a constant proportional to W. Step 4 can be achieved by adding unordered neighbors of the chosen v_{i^*} to N, and then updating $Attract(i)$ for each $v_i \in N$. For a given v_i, we can generally update $Attract(i)$ by adding the attraction exerted by v_{i^*} and subtracting the attraction exerted by $v_{\pi_{index-W}}$, the vertex just exiting the new window. Hence, WINDOW runs in $O(nW)$ time when $T = 1$; if $T > 1$, then Step 4 requires $O(T)$ time and the overall complexity becomes $O(nWT)$.

The WINDOW Algorithm

Input: Netlist $H(V, E)$
S $\equiv$ set of ordered vertices
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 $v_i^* \in V$ and set $\pi(1) = i*$. **for** each $v_i \in V - S$ compute $Attract(i)$.
2. **for** $index = 2$ **to** n **do**
3. Choose $v_{i^*} \in V - S$ such that $Attract(i^*)$ is optimal. Set $\pi(index) = i^*$
4. **for** each $v_i \in V - S$ update $Attract(i)$ such that
   if $index - W + 1 \leq j \leq index$, **then** v_{π_j} has full attraction on v_i
   if $index - W - T + 1 \leq j \leq index - W$ **then** v_{π_j} has $\frac{index-W-i}{T}$% attraction on v_i
5. **return** $v_{\pi_1}, v_{\pi_2}, \ldots, v_{\pi_n}$

Figure 4: The WINDOW Algorithm.

5 Experimental Results

The "DP-RP" (Dynamic Programming/Restricted Partitioning) algorithm [2] can be used to construct a clustering from a vertex ordering. The algorithm accepts a netlist and a vertex ordering as input and returns a *restricted partitioning*, i.e., a k-way clustering 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 clustering; this is possible for any clustering objective that is a *monotone* function of an intercluster cost metric (e.g., Absorption and Scaled Cost). Although the complexity of DP-RP depends on the objective function, $O(nU + kn(U - L))$ implementations have been given for Scaled Cost [2] and Absorption [1]. In what follows, we use DP-RP to find k-way clusterings from WINDOW derived orderings. A complete description of the DP-RP algorithm is given in the appendix.

5.1 WINDOW Versus Previous Clustering Methods

Table 1 compares DP-RP clusterings derived from WINDOW vertex orderings with the MBC [4], RW-ST [12], AGG [3], and ANNEAL [22] clusterings, in terms of the Scaled Cost, Absorption, DS, and two-phase FM min-cut bisection objectives.[5] The clusterings for MBC, RW-ST, and AGG were obtained from [12] and [3] and we implemented ANNEAL using the same temperature schedule as [22]. Below each MCNC benchmark, we report the number modules, nets, and clusters; the same number of clusters is used as in the experiments of [12, 3]. For each algorithm, we conducted 20 FM runs with and without the corresponding module areas, though none of the algorithms use this information (no area information was available for biomed or industry2). The FM cuts for AGG are given for 200 runs since 10 clusterings were available, and we report the best Scaled Cost and Absorption values over the ten AGG clusterings available for each test case.

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.[6] (Separately, we have found that these parameters are not optimal, and that in particular the tradeoffs between W and T remain unclear [1].) For Scaled Cost and Absorption, we used the attraction functions described in Section 3. Notice that the WINDOW results for Scaled Cost and Absorption correspond to different clusterings. As one would expect, when WINDOW optimizes one objective, the clustering usually worsens with respect to other objectives. For example, with the primary1 test case, minimizing Scaled Cost (173.1) leads to Absorption = 621.9, while maximizing Absorption (687.6) leads to Scaled Cost = 234.8. The FM results for WINDOW are reported using the Absorption clusters, though the Scaled Cost cuts were just as good.

[5] See Footnote 2 for the definition of the DS objective. Because computing DS requires $O(n^3)$ time, we report only the DS value for the AGG clustering with lowest Scaled Cost, and we omit computing DS for the biomed and industry2 clusterings. Since DS is monotone, DP-RP can be used to split an ordering according to this objective; however, we did not devise an attraction function for DS, and simply used the Scaled Cost attraction for WINDOW orderings. Despite the fact that we did not directly optimize DS, WINDOW still yielded an average of 8.3% improvement over the combined results of the previous algorithms.

[6] The *eccentricity* of a vertex v is the distance of the farthest graph vertex u 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. Ideally, the starting vertex for the ordering should be an endpoint of a diameter of the circuit, since the ordering should begin in a low-density region, move through the dense part of the circuit, and end in a low-density region (i.e., the other endpoint of the diameter). Computing a diameter is computationally expensive, so we compute a pseudo-peripheral vertex in linear time; such a vertex is likely to be a diameter endpoint and thus will be contained in a low-density region of the circuit.

Test Case	Algorithm	SCost	Absorp	DS	FM Cuts with Area			FM Cuts Unit Size		
					Min	Max	Avg	Min	Max	Avg
prim1-GA	WINDOW	173.1	687.6	1.471	55	85	74.5	60	92	75.2
833	RW-ST	287.9	629.9	1.325	57	95	75.0	53	85	68.7
(902)	AGG	277.9	437.0	0.879	52	97	73.8	53	107	73.4
[191]	MBC	254.0	309.3	1.258	48	97	73.0	55	104	77.4
	ANNEAL	304.1	537.0	0.802	58	91	76.5	71	91	74.8
prim2-GA	WINDOW	57.69	2257	1.539	229	318	262.0	258	258	258.0
3014	RW-ST	82.81	2013	1.566	240	240	240.0	258	258	258.0
(3029)	AGG	89.73	1227	1.048	146	352	242.6	258	258	258.0
[702]	MBC	82.44	736.4	1.238	240	240	240.0	222	222	222.0
	ANNEAL	88.70	1844	0.882	240	240	240.0	258	258	258.0
test02	WINDOW	97.02	1327	1.662	42	129	93.2	105	134	112.0
1663	RW-ST	150.7	1123	1.593	42	116	98.0	99	132	124.3
(1720)	AGG	164.7	706.2	0.657	42	173	103.0	97	189	138.4
[445]	MBC	137.4	407.0	1.231	42	42	42.0	136	177	154.1
	ANNEAL	150.5	1037	1.025	42	143	75.7	135	172	150.5
test03	WINDOW	91.50	1247	1.736	68	96	85.2	67	143	107.5
1607	RW-ST	156.2	1101	1.566	71	109	84.8	60	108	79.5
(1618)	AGG	153.9	678.4	1.204	50	125	86.9	68	153	102.7
[327]	MBC	140.7	379.5	1.185	59	149	87.2	61	146	87.8
	ANNEAL	153.4	980.8	0.785	50	136	97.5	95	132	108.5
test04	WINDOW	100.2	1303	2.014	20	20	20.0	61	120	83.3
1515	RW-ST	151.8	1181	1.879	14	14	14.0	56	104	87.2
(1658)	AGG	193.3	833.9	1.135	13	74	22.8	58	142	101.1
[424]	MBC	160.3	415.5	1.297	20	20	20.0	84	171	111.3
	ANNEAL	164.7	1062	1.305	20	20	20.0	93	121	105.0
test05	WINDOW	55.28	2279	1.831	45	138	67.4	101	148	128.2
2595	RW-ST	88.52	2051	1.689	41	131	81.3	108	160	136.3
(2750)	AGG	103.0	1527	1.262	30	159	73.7	93	187	145.5
[424]	MBC	90.08	680.7	1.275	37	109	59.2	147	213	188.5
	ANNEAL	90.70	1834	0.933	46	85	63.8	124	176	149.9
test06	WINDOW	106.9	1274	1.516	75	106	92.2	70	102	84.3
1752	RW-ST	178.7	979.2	1.367	76	110	95.2	71	114	93.8
(1541)	AGG	163.2	359.0	1.183	68	118	85.7	68	121	88.7
[476]	MBC	142.4	315.3	1.331	83	125	110.5	72	124	105.1
	ANNEAL	160.4	936.9	1.034	77	114	91.4	77	120	100.0
19ks	WINDOW	47.51	2556	1.883	143	172	151.1	136	196	170.2
2844	RW-ST	81.08	2395	1.578	152	229	162.9	149	171	158.4
(3282)	AGG	86.50	1485	1.022	124	258	179.7	121	279	181.9
[737]	MBC	75.50	719.9	1.166	120	204	173.0	127	223	176.3
	ANNEAL	80.05	2124	0.534	148	231	177.4	148	221	191.3
bm1	WINDOW	137.8	692.5	1.278	61	86	74.8	70	85	76.5
882	RW-ST	258.5	637.9	1.221	56	83	65.5	61	81	70.3
(903)	AGG	266.0	426.6	0.813	47	93	68.9	52	94	69.2
[216]	MBC	340.5	199.1	1.189	59	88	78.3	52	89	75.3
	ANNEAL	241.0	535.8	1.105	58	108	85.7	56	95	71.8
biomed	WINDOW	20.91	5070					164	220	200.2
6514	ANNEAL	32.27	4129					149	265	205.6
industry2	WINDOW	11.35	10747					392	726	536.7
12637	ANNEAL	20.00	8253					533	1046	786.6

Table 1: Comparison between WINDOW and four other clustering algorithms. The numbers below each test case are the number of modules, number of nets (in parentheses) and the number of clusters (in brackets). Biomed and industry2 respectively have 5742 and 13,419 nets and 1303 and 2527 clusters. CPU times for our methodology on a Sun Sparc-10 were 9.7, 36 and 63 seconds 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.

For Scaled Cost and Absorption, WINDOW clusterings averaged 34.2% and 13.2% respective improvement versus the closest other results. For two-phase FM results are mixed, with all the algorithms being fairly comparable. Considering how well WINDOW performed in terms of the three clustering objectives, there seems to be little correlation between any of the objectives and the "meta-objective", two-phase FM. We believe that the most important direction for future research in clustering is to understand the characteristics of clusterings that perform well in terms of meta-objectives.

5.2 WINDOW Versus Other Vertex Orderings

We also compared WINDOW orderings to seven other vertex ordering constructions: DFS, BFS, King, Cuthill-McKee (Cut-McK) [19], Max-Adjacency (Max-Adj) [17], EIG1 [13], and SFC [2]. 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. Table 2 provides the Scaled Cost and Absorption values for the clusterings generated by DP-RP. SFC results are the best results obtained from ten SFC orderings.

For each test case, 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 an 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 six ordering constructions. Our results suggest that the local traversals (DFS, BFS, etc.) do not capture the objective or the constraints of the clustering problem and the "global" EIG1 and SFC methods cannot make the local decisions necessary for DP-RP to generate a good clustering for large k. We also note that in a separate set of experiments, we used DP-RP to generate relatively balanced 5-way clusterings for WINDOW, EIG1, and SFC, and WINDOW performed best [1]. This result suggests that EIG1 and SFC cannot adapt to fixed size bounds, even on a global level.

In conclusion, we have developed a general framework for constructing vertex orderings, and explored its applications to circuit clustering. By setting an appropriate attraction function and window size, we obtained superior clusterings for a variety of clustering objectives in the literature.

Clustering Objective	Test Case	Ordering Algorithm DFS	BFS	King	Cut-McK	Max-Adj	EIG1	SFC	WINDOW
Scaled Cost	prim1	222.1	224.5	209.6	226.7	176.1	244.1	204.7	173.1
	prim2	69.47	74.73	68.97	73.24	61.08	78.82	68.10	57.69
	test02	121.4	131.7	118.1	129.6	100.3	137.4	131.4	97.02
	test03	124.0	134.4	119.6	130.1	97.28	132.1	126.5	91.50
	test04	137.3	145.7	132.3	143.9	109.2	155.4	147.7	100.2
	test05	76.8	85.4	73.27	83.97	60.42	85.08	76.06	55.28
	test06	123.3	129.3	121.8	127.5	107.0	131.5	142.8	106.9
	19ks	55.0	67.2	63.12	69.79	50.18	73.58	66.37	47.51
	bm1	175.5	183.2	168.3	185.9	132.8	184.8	146.2	137.8
	biomed	26.41	28.23	26.95	28.99	21.51	32.53	22.98	20.91
	indstr2	14.25	15.21	14.28	15.39	11.73	16.95	13.25	11.35
Absorption	prim1	426.9	278.6	489.0	259.6	534.6	312.8	513.3	687.6
	prim2	1065	654.4	1209	598.6	1355	600.2	1114	2257
	test02	599.1	256.0	709.8	282.9	991.4	468.8	618.6	1327
	test03	578.2	253.3	649.3	300.6	951.7	525.5	640.3	1247
	test04	585.6	332.2	716.8	326.8	992.3	382.6	625.3	1303
	test05	927.0	459.8	1069	490.8	1562	891.9	1324	2279
	test06	439.3	234.5	546.5	247.6	765.3	264.9	349.1	1274
	19ks	1626	981.5	1389	636.2	2042	806.2	1573	2556
	bm1	449.5	281.4	512.8	278.0	546.4	355.5	513.8	692.5
	biomed	1833	1470	2134	1427	3608	491.1	3403	5070
	indstr2	4066	1951	4392	1774	7551	1987	4630	10747

Table 2: Scaled Cost and Absorption values of the clusterings DP-RP generated from the six given ordering constructions. Clustering sizes were generated for $L = 1$ and $U = 20$ using DP-RP with k same as in Table 1.

We leave open the question of finding improved attraction functions and size constraints for meta-objectives that cannot be defined explicitly, such as two-phase FM enhancement.

Acknowledgements

We thank Dr. Lars Hagen for supplying FM code and the RW-ST and BUI clusterings, and we thank the anonymous reviewers for their helpful comments.

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, "Multi-way Partitioning Via Spacefilling Curves and Dynamic Programming", *Proc. ACM/IEEE Design Automation Conf.*, 1994, pp. 652-657 (extended version to appear in *IEEE Trans. on CAD*).

[3] 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.

[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”, *IEEE Trans. on CAD* 13(9), 1994, pp. 1088-1096.

[6] N. C. Chou, L. T. Liu, C. K. Cheng, W. J. Dai, R. Lindelof, “Circuit Partitioning for Huge Logic Emulation Systems”, *Proc. ACM/IEEE Design Automation Conf.* 1994, pp. 244-249.

[7] 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.

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

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

[10] T. Fujii, H. Horikawa, T. Kikuno, and N. Yoshida, “A Heuristic Algorithm for Gate Assignment in One-Dimensional Array Approach”, *IEEE Trans. on CAD* 6(2), 1987, pp. 159-164.

[11] J. Garbers, H. J. Promel and A. Steger, “Finding Clusters in VLSI Circuits” *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1990, pp. 520-523.

[12] L. Hagen and A. B. Kahng, “A New Approach to Effective Circuit Clustering”, *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1992, pp. 422-427.

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

[14] K.M. Hall, “An r-dimensional Quadratic Placement Algorithm”, *Manag. Sci.*, 17, 1970, pp. 219-229.

[15] B. W. Kernighan and S. Lin, “An Efficient Heuristic for Partitioning Graphs”, *Bell Syst. Tech. J.* 49(2), 1970, pp. 291-307.

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

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

[18] D. Peleg and A. Schaffer, “Graph Spanners”, *J. of Graph Theory* 13, 1989, pp. 99-116.

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

[20] 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.

[21] L.A. Sanchis, “Multiple-way Network Partitioning”, *IEEE Trans. on Computers*, 38, 1989, pp. 62-81.

[22] W. Sun and C. Sechen, “Efficient and Effective Placements for Very Large Circuits” *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1993, pp. 170-177.

DP-RP Algorithm

Input: Hypergraph $H(V, E)$
Vertex Ordering $\{v_{\pi_1}, v_{\pi_2}, \ldots, v_{\pi_n}\}$
L, U $\equiv$ Lower and upper cluster size bounds
k $\equiv$ Number of clusters

Output: P^k $\equiv$ Restricted k-way partitioning solution

Vars: k' $\equiv$ Index denoting current partitioning size
$l+1$ $\equiv$ Beginning index of possible new cluster

1. $\forall i, j$ with $L \leq j - i + 1 \leq U$ compute $w(C_{[i,j]})$ using **Cluster_Costs**
2. **for** $k' = 2$ **to** k **do**
3. **for** $j = 1$ **to** n **do**
4. Find the best k'-way partitioning for $\{v_{\pi_1}, v_{\pi_2}, \ldots, v_{\pi_j}\}$ composed of the best $(k'-1)$-way partitioning for $\{v_{\pi_1}, v_{\pi_2}, \ldots, v_{\pi_l}\}$ and cluster $C_{[l+1,j]}$ for $j - U \leq l \leq j - L$
5. **return** P^k = the k-way partitioning derived for $\{v_{\pi_1}, v_{\pi_2}, \ldots, v_{\pi_n}\}$

Figure 5: The DP-RP algorithm of [2].

Appendix: The DP-RP Algorithm

The restricted partitioning formulation requires that each cluster is a contiguous subset of the vertex ordering. Thus, every such cluster may be uniquely denoted by $C_{[i,j]} = \{v_{\pi_i}, v_{\pi_{i+1}}, \ldots, v_{\pi_j}\}$, where $L \leq j - i + 1 \leq U$. The number of clusters that need to be considered as part of a solution P^k is bounded by nU. The DP-RP algorithm (see Figure 5) first computes $w(C_{[i,j]})$ for each possible cluster through a Cluster_Costs procedure (which depends on the objective). Observe that the cluster $C_{[1,j]}$ is the unique 1-way clustering solution over $\{v_{\pi_1}, v_{\pi_2}, \ldots, v_{\pi_j}\}$. We inductively assume that for each j, we have computed $\hat{P}^{k'-1}_{[1,j]}$, the optimal $(k'-1)$-way clustering solution over $\{v_{\pi_1}, v_{\pi_2}, \ldots, v_{\pi_j}\}$. By considering all indices l, such that $j - U \leq l \leq j - L$, we can find an optimal k'-way clustering of the form $\hat{P}^{k'}_{[1,j]} = \hat{P}^{k'-1}_{[1,l]} \cup C_{[l+1,j]}$. A sufficient condition for DP-RP to generate optimal solutions is for f to be *monotone* over w: for any $P^k = \{C_1, C_2, \ldots, C_k\}$ and $Q^k = \{C'_1, C'_2, \ldots, C'_k\}$ with $w(C_i) \leq w(C'_i)$ for $1 \leq i \leq k$, f is monotone nondecreasing if and only if $f(P^k) \leq f(Q^k)$. From the definitions in Section 3, it is easy to see that Absorption, Scaled Cost, and DS are all monotone objectives.