Studies of Clustering Objectives and Heuristics for Improved Standard-Cell Placement

Andrew B. Kahng and Rahul Sharma
UCLA Computer Science Dept., Los Angeles, CA 90095-1596 USA

Abstract

This paper describes ongoing studies of clustering objectives and heuristics, along with their effect on top-down partitioning based standard-cell placement. Clustering for placement has three main facets – the objective, the heuristic, and the benefits – but the connections among these facets have never been clarified. Our studies represent the first steps toward reconciling these three facets. Specifically: (1) we consider seven distinct objectives (three from the literature, four new), by which we can rank given clustering heuristics; (2) we consider 35 variants of 13 clustering heuristics (four from the literature, nine new); (3) we evaluate the benefits of given clusterings using a recent top-down partitioning based placement tool [HK97] whose approach and solution quality reflect those of leading-edge tools (e.g., quadratic top-down placers); and (4) based on these steps, we identify (new) objectives and heuristics that seem most beneficial to placement. Our paper concludes by discussing some limitations and near-term extensions to the present study.

1 Preliminaries: Clustering for Placement

A *clustering* of a standard-cell netlist groups cells into disjoint *clusters*. With respect to the placement phase of physical design, there are at least three major purposes for clustering. First, clustering contracts a large problem instance into a smaller instance; this saves runtime or allows better search of the solution space, and is the primary motivation in the literature. Second, when it is known that cells should be near each other in the placement, clustering them together prevents, e.g., a top-down partitioner from making a mistake. Third, clustering can incorporate knowledge of problem structure that the placer would otherwise have to ignore (e.g., recognition of synthesized buffer-inverter trees, recognition of two-dimensional regularity [NJ96], "electrical clustering", or hierarchy-based clustering [BHB96]); we believe that this will be the key motivation for clustering in the future. In our work, we study clustering for placement in terms of three elements: the clustering objective, the clustering heuristic, and the perceived benefit to placement.

- **The Clustering Objective.** The true definition of a good clustering is "one that leads to better placement solution quality". As noted in [AK94], this is a *meta-objective*, e.g., one can replace "placement" by "two-phase FM partitioning" in the definition (see also comments by Soukup regarding placement objectives [Sou81]); it cannot be optimized by traditional methods. The key question is whether a good clustering objective, optimizable by traditional methods, can be *deduced* from knowledge of the placer that uses the clustering. So far, only intuitive clustering objectives have been proposed,

most (the Absorption objective of [SS93] being the exception) having no clear connection to placement. Since leading-edge (global) placers are based on top-down partitioning, certain partitioning objectives arguably have connections to placement; such objectives include D/S ratio [HK92] and Scaled Cost [CSZ93] (see [AK95] for a survey).

Our eventual goal is to *deduce* a clustering objective (i.e., an analytic quality measure of a given clustering) from a realistic model of (top-down) cell placement. Such an objective should be efficiently optimizable, and lead to clusterings that consistently improve the placement solution quality. For now, we evaluate a number of existing objectives (Scaled Cost, D/S Ratio, Absorption), as well as new several new placement-specific clustering criteria (e.g., whether the clustering identifies groups of cells that end up together in a high-quality final placement). These criteria are summarized in Section 2 below. We assess these various criteria according to whether they reflect with high fidelity the benefits of clustering to placement. In other words, we see whether a "better" clustering according to a given criterion can be expected to yield a better placement solution.[1]

- **The Clustering Heuristic.**

  Netlist clustering heuristics are usually classified as either *top-down* or *bottom-up*. A top-down heuristic recursively partitions the netlist into subclusters, e.g., using a ratio-cut objective [WC91]. A bottom-up heuristic initially assigns each cell to its own cluster and agglomeratively merges clusters into larger clusters, e.g., by random matching [BCLS87] [BHJL89]. Traditionally, top-down approaches gain from available "global" information (hierarchy, timing, other structure) but suffer from having to perform global analysis of the netlist. (For example, a top-down partitioning based clustering approach might do as much work as the top-down partitioning based placer that it is trying to assist.) Symmetrically, bottom-up approaches are faster but can make myopic mistakes. This simple picture has two problems: (1) many approaches such as local search (e.g., by annealing [SS93]) or seeded cluster growth (e.g., for regularity extraction [NJ96]) are neither top-down nor bottom-up, and (2) the distinction between "local" and "global" information is blurred when one considers the "global" information available from cell names, simple netlist transformations (e.g., sorts that can bias later tie-breaking), etc.

  Our study avoids true top-down heuristics because of their complexity, and because they are largly subsumed by our placement testbed (the QUAD placer of [HK97]). All other clustering heuristics are considered. In Section 3, we review clustering heuristics that have been claimed to improve placement performance including MFFC-based [CX95], annealing for maximum Absorption [SS93], and regularity extraction [NJ96]; we subsume the cone-based approach of [TL95] by an MFFC variant. We also review heuristics that have been claimed to improve two-phase FM bipartitioning (matching-based [BCLS87]

[1] We are not yet satisfied with the rigor of the "deduction process" that leads to our new criteria, and our new criteria in general are not optimizable.

[BHJL89] and Scaled Cost based WINDOW [AK94]). We then describe several new clustering heuristics, including three variants of MFFC clustering (MFFC-LA, IMFFC, IMFFC-LA) and six others that we call Alg1, Alg2, Alg3, Alg4, Alg5 and Alg6. Most of these clustering heuristics are controllable with respect to either maximum cluster size or average cluster size (the latter is equivalent to a target number of clusters). In all, for each test circuit we obtain over 35 different clusterings from 13 distinct heuristics. While correspondences between the heuristics of Section 3 and the objectives of Section 2 are yet unsatisfactory (reflecting the existing literature), our range of clusterings does represent existing approaches.

- **The Benefit to Placement.** From the purposes listed above, we may discern two possible benefits from clustering: (1) clustering can speed up the placement process, or (2) clustering can lead to better placement solution quality. Making these ideas more concrete requires a model of the placement process.

  Our work is aimed at benefiting a global placer, i.e., a fast tool that constructs an "initial" placement based on simplified models for cell layouts, routing estimates, and placement objective. (We assume a subsequent detailed (annealing-based) placement phase that optimizes pin alignments, cell orientations, detailed congestion, etc.) We assume that the global placer is based on top-down quadrisection or bisection. This reflects the approach actually used in modern "quadratic" placers (PROUD, [TKH88], GORDIAN/GORDIAN-L [KSJ88] [SDJ91] (to a limited extent), and various commercial tools), where a partitioner is "seeded" from the one-dimensional solution of the linear system induced by a quadratic wirelength objective. Top-down quadrisection is also the approach used in our testbed placer [HK97]; as reported in [Hua97], this tool outperforms GORDIAN-L/DOMINO [DJS94] and is on par with leading commercial tools. We assume the traditional minimum spanning tree length objective; timing objectives are ignored in this stage of our work. We also ignore runtime issues (as discussed in Section 4, we can invoke the placer in a mode where clustering cannot increase runtime).

The three facets of clustering-based placement – the objective, the heuristic, and the intended purpose with respect to a given placer – are completely intertwined. Ideally, a given desired effect on a given placer will determine the clustering objective; a given objective will determine the clustering heuristic; and any given heuristic will exhibit a certain effect on the placer. In the existing literature, links from placer to objective, or from objective to heuristic, are non-existent; our work is a first step toward reconciling the three facets of clustering for placement. The remainder of the paper is organized as follows. Section 2 describes seven distinct objectives (three from the literature, four new), according to which we rank given clustering heuristics. Section 3 describes 35 variants of 13 clustering heuristics (four from the literature, nine new), and Section 4 evaluates the benefits of given clusterings using a recent top-down partitioning based placement tool [HK97] whose approach and solution quality reflect those of leading-edge tools. Based on these experiments,

we identify (new) objectives and heuristics that seem most beneficial to placement. Section 5 concludes the paper by discussing known limitations, as well as ongoing or near-term extensions, to the present study.

Notation

We now establish some basic notation. A VLSI netlist hypergraph $N(V, E)$ has $V = \{v_1, v_2, \ldots, v_n\}$ and $E = \{e_1, e_2, \ldots, e_m\}$, with the n vertices corresponding to cells and the m hyperedges corresponding to signal nets. Cells are of two types: *sequential* (storage elements) and *combinational*; this distinction is useful for performance-driven placement. We also distinguish *core* cells from IO (*pad*) cells; the latter are either *primary inputs* (PIs) or *primary outputs* (POs). Each (directed) hyperedge $e \in E$ is a subset of V ($|e| \geq 2$) containing one *source* vertex $S(e)$, with the remaining vertices of the hyperedge being the set of *sinks* $D(e)$. For each cell v, the set of hyperedges *incident* to v is denoted by $N(v) = \{e \in E \mid v \in e\}$.

A clustering of the netlist into k disjoint clusters (subsets of V) is denoted $P^k = \{C_1, C_2, \ldots, C_k\}$, with $V = \cup_{h=1}^{k}\{C_h\}$. A given objective to be optimized is denoted $F(P^k)$, i.e., it is a function of the clustering solution. The set of hyperedges *cut* by a cluster C is given by $E(C) = \{e \in E \; s.t. \; 0 < |e \cap C| < |e|\}$. The set of hyperedges cut by a clustering solution P^k is $E(P^k) = \cup_{h=1}^{k} E(C_h)$.

In discussing clustering heuristics and objectives, certain circuit-structural terms are convenient. A given source is a *fanin* to each of its sinks; a given sink is a *fanout* of each of its sources. We use the term *input set* of a cell v, IS_v, to denote the set of cells that are fanins to cell v. Similarly, the *output set* of v, OS_v, denotes the set of cells that are fanouts of cell v. The *input latch set* of a cell v, IL_v, is the set of sequential cells that have paths to cell v consisting of only combinational cells (*combinational paths*). The *output latch set* of v, OL_v, is the set of sequential cells that can be reached from cell v through combinational paths. Finally, a *Cone* of a cell v, denoted $Cone_v$, is a subhypergraph of cells consisting of v and its predecessors such that any path from a vertex in $Cone_v$ to v lies entirely in $Cone_v$. We call v the *root* of $Cone_v$, i.e., $v = root(Cone_v)$.

2 Clustering Objectives

We have studied seven clustering objectives. Three of these, Scaled Cost, DS Ratio and Absorption, are well known in the literature [AK95].[2]

- **Scaled Cost.** To integrate cutsize and cluster size balance within a single objective, Chan et al. [CSZ93] proposed the *Scaled Cost* objective for multi-way netlist partitioning.

  Minimum Scaled Cost: Minimize

$$F(P^k) = \frac{1}{n(k-1)} \sum_{h=1}^{k} \frac{|E(C_h)|}{w(C_h)}.$$

[2]While many works have studied netlist clustering for various purposes, Scaled Cost, DS Ratio and Absorption are among the very few explicit analytical *objectives* that have been proposed. Many clustering heuristics have been described in the literature without any associated objective; these are discussed in Section 4.

This is a k-way generalization of the ratio cut objective, and has been optimized in other works, e.g., the Window clustering approach of Alpert and Kahng [AK94].

- **DS Ratio.** The *DS Ratio* objective [HK92] is:

  Maximum DS Ratio: Maximize

$$F(P^k) = \frac{1}{n} \sum_{h=1}^{k} \frac{degree(C_h)}{separation(C_h)}$$

  where $degree(C_h)$ is the average number of nets incident to each cell of the cluster that have at least two pins in the cluster, and $separation(C_h)$ is the average length of a shortest path between two cells in C_h ($= \infty$ if the cluster is disconnected). This objective was originally proposed in the context of random-walk based clustering to improve two-phase Fiduccia-Mattheyses netlist bipartitioning. Because evaluating the DS ratio can have up to cubic time complexity, the objective is more useful for comparison than for optimization of clustering solutions.

- **Absorption.** The *Absorption* objective [SS93] measures the sum of the fractions of nets "absorbed" by the clusters:

  Maximum Absorption: Maximize

$$F(P^k) = \sum_{h=1}^{k} \sum_{e \in E | e \cap C_h \neq \emptyset} \frac{|e \cap C_h| - 1}{|e| - 1}$$

  e.g., net e incident to cluster C_h adds absorption zero if it has only one cell in C_h, and adds absorption one if all of its cells are in C_h. Absorption was proposed specifically for standard-cell placement; Sun and Sechen use a simulated annealing heuristic to optimize the clustering, as detailed in Section 4.

The four remaining clustering objectives, which we call Split, Density, C and C', were developed during the course of our work. Our initial goal, seen in the Split and Density criteria, was to "reverse-engineer" a useful clustering objective from knowledge of how the assumed top-down partitioning based placement tool works.

- **Split.** The Split objective evaluates a given clustering with respect to a "good" netlist bisection (that is, a heuristic minimum-cut partitioning of V into two equal-area partitions V_1 and V_2). For each cluster C_i with $|C_i| > 1$, Split determines whether it is split by the bisection, i.e., whether $C_i \cap V_1$ and $C_i \cap V_2$ are both nonempty. A better clustering will have fewer clusters split by the netlist bisection. This clustering criterion exactly reflects the motivations for top-down min cut placement: wiring area is a function of cutsize, and strongly-related cells should be near each other in the placement.[3] Table

[3] We use the minimum-cut criterion for several reasons: (1) it is a standard objective in leading top-down placement tools, including quadratic placers and our testbed placer; (2) additional components of the placement objective (e.g., to reflect timing

1 below gives an example of our Split analyses for the ALG1 heuristic clustering: we separately track the proportions of split clusters for clusters having sizes of 2-4, 5-8, 9-16 and ≥ 17 cells, as well as for all clusters of size ≥ 2 cells.

Benchmark	#cells	#single	# 2-4	# 5-8	# 9-16	# 17+	# ≥ 2
Biomed	6514	998	709/8	55/6	122/14	79/13	965/41
Avqsmall	21918	4668	1450/7	437/1	355/15	133/22	2375/45
Avqlarge	25178	4582	1385/6	380/0	475/2	221/21	2461/29

Table 1: Split analysis for the ALG1 heuristic clustering. Each entry of form x/y indicates the number of clusters in a given size range, and the number of split clusters in that size range.

- **Density.** The Density objective evaluates a given clustering with respect to a "good" placement of the netlist. For each cluster C_i with $|C_i| > 1$, Density measures (1) the total cell area in the cluster, and (2) the area of the smallest bounding box enclosing all placed locations of cells in the cluster, then computes the ratio of these two values. A ratio approaching 1 indicates that the cluster's cells remained close together even until the end of the top-down placement process; a ratio close to 0 indicates that the placer did not keep the cluster's cells together.[4] Table 2 below gives an example (with respect to a QUAD placement) of our Density analyses for the ALG1 heuristic clustering: we separately track the average density ratios for clusters having sizes of 2-4, 5-8, 9-16 and ≥ 17 cells, as well as for all clusters of ≥ 2 cells.

Benchmark	#cells	#single	2-4	5-8	9-16	17+	≥ 2
Biomed	6514	998	0.24	0.13	0.08	0.04	0.21
Avqsmall	21918	4668	0.38	0.19	0.19	0.05	0.30
Avqlarge	25178	4582	0.40	0.26	0.19	0.07	0.31

Table 2: Density analysis for the ALG1 heuristic clustering. Number of clusters of each size range is the same as in Table 1.

The main weakness of Split and Density is that each is evaluated according to a "snapshot" of the top-down placement process. For Split, the evaluation is with respect to only the initial top-level partition, and is therefore independent of subsequent steps in the top-down process. (Intuitively, one would prefer (1) an objective function that is sensitive to whether a cluster is first split at the third, fourth, tenth, etc. partitioning level, and (2) an objective that is sensitive to further splits of a cluster in later partitioning steps.) For Density, the evaluation is with respect to a final placement solution, and is independent even of the fact that a top-down approach was used to derive this solution. Thus, we have developed two additional clustering criteria, C and C', that are more sensitive to the top-down partitioning based placement *process*.

criticalities) can usually be modeled via (hyper)edge weights, so that minimum-cut heuristics still apply; and (3) several clustering heuristics plausibly minimize nets cut. In our study, a "good" bisection is obtained from the multilevel FM code of Alpert et al.; this code typically achieves best-known results for the standard public benchmark sets [AHK96] [Alp96] and is also used within our testbed placer. The code can be run in a randomized mode; we evaluate Split with respect to the best 5 out of 20 executions, then report average numbers of split clusters.

[4] In our study, "good" placements were obtained from the testbed placer QUAD, with optional heuristics for cycling and overlapping of partitioned regions to improve solution quality. We also analyzed placements from the commercial TimberWolf placer [Swa96], and from the GORDIAN-L/DOMINO and SPEED placers [Rie95], as reported in Section 4.

- **The C Objective.** The C objective is given by

$$C = k \cdot 4^L + \sum_{i=1}^{L} n_i \cdot 4^{L-i+1}$$

where L is the total number of partitioning levels in the placement, k is the total number of clusters in the clustering, and n_i is the total number of cells belonging to clusters that are first split at level i. This objective assesses larger "penalties" the earlier a cluster is broken by the partitioning; better clusterings have smaller C value. The objective also assumes top-down quadrisection (hence the constant "4"), with the first partitioning level corresponding to $i = 1$. The term $k \cdot 4^L$ penalizes clustering schemes that have too many single-cell clusters. The C objective performs reasonably for extreme cases,[5] but has several failings including its assumption of quadrisection at every level.[6]

- **The C' Objective.** The C' objective is a refinement of the C objective, notably in that it is sensitive to whether 2- or 4-way partitioning was applied at a given level. Also, whereas C penalizes an early split of a cluster, C' increases the "credit" received by a cluster as it survives each additional partitioning operation: better clusterings have larger C' value. More formally,

$$C' = \frac{1}{k} \sum_{i=1}^{n} \sum_{j=1}^{L} (CellLevArray_{ij})^j$$

where n is the number of cells in the netlist, k is the number of clusters, and L is the number of partitioning levels. The value of $CellLevArray_{ij}$ is set to 2 (resp. 4) if the level-j partition containing *cell* v_i was created by a 2-way (resp. 4-way) partition. However, if the cluster containing cell v_i has only one cell, then values of $CellLevArray_{ij} = 0$ for all j and are ignored in the summation. Furthermore, if the cluster containing cell v_i is split at some level j_0, then values of $CellLevArray_{ij} = 0$ for all $j \geq j_0$ and are also ignored in the summation. The C' measure also gives reasonable results for the trivial extreme cases,[7] but it too has inconsistencies.[8]

Notice that Split, Density, C and C' are not optimizable "objectives". Rather, their evaluation of a clustering *depends on a particular execution* of either a partitioner or a placer. At this stage of our work, we can only hope to demonstrate that certain of our new criteria (particularly C' or variants) can identify promising clustering heuristics.

[5] When all cells are in a single cluster, $C = (n+1) \cdot 4^L$ where n is the total number of cells in the netlist. When each cell is in its own cluster, $C = n \cdot 4^L$). It seems reasonable that the value of C is similar for these two "trivial" clustering solutions.

[6] In top-down placement, quadrisection levels are limited by the number of rows: typically, the placer will revert to bisection levels once it begins partitioning within single rows. Furthermore, because the number of rows and the number of cells per row are not perfect powers of 2, two clusters that are intact at a given "level" may actually have survived very different sequences of partitioning operations.

[7] When all cells are in a single cluster, $C' = 0$ because $CellLevArray_{ij} = 0$ for all cell indices i and for all levels $j \geq 1$. When each cell is in its own cluster, again $C' = 0$ because $CellLevArray_{ij} = 0$ for all j and all single-cell clusters.

[8] For example, C' fails to keep track of the cells in a cluster after it has been split. Ideally, more credit (less negative credit) should be given to a cluster which is split but remains clumped in only a few partitions than a cluster whose cells are scatterred randomly. Also, interchanging the order in which we apply 4-way/2-way partitioning results in different C' values even though the final result of partitioning is unaltered.

3 Clustering Heuristics

In this section, we review clustering heuristics from the literature, as well as new heuristics developed for this study.

- **Matching-based Clustering.** A *matching* in the netlist is a set of disjoint pairs of cells, such that each pair shares a common net. Bui et al. [BCLS87] [BHJL89] find a random maximal matching and merge each pair of cells into a cluster: this will ideally result in a set of $\frac{n}{2}$ clusters; matchings can be iteratively computed on the contracted netlist, yielding a hierarchy of clusterings with size $\frac{n}{4}$, $\frac{n}{8}$, etc. A similar approach was proposed earlier by [GB83]. The "heavy-edge matching" variant of Karypis and Kumar [KK95] is used in the multilevel partitioning engine within our testbed placer, QUAD [Alp96] [HK97]. All existing matching-based variants were originally proposed in the context of (hyper)graph partitioning.

- **Annealing for Maximum Absorption.** Sun and Sechen [SS93] believe that clusterings that maximize the Absorption objective are best suited to the TimberWolf simulated annealing based placer. In [SS93], the clustering process itself uses simulated annealing, with neighborhood structure induced by single-cell moves from one cluster to another. The placement neighborhood structure of [SS93] is induced by cluster swapping (the clusters are treated as large cells, with a given cluster's contents placeable only in a single-height row or a double-height row); cluster size lower and upper bounds are established relative to a desired average cluster size. This is because too much variation in cluster size may make the placement solution infeasible.

- **Cone-Based Variants.** Cong and Ding [CD93] find maximum "fanout-free cones" (single-output subnetworks), or MFFCs, in the DAG of a combinational network; they apply this decomposition to FPGA technology mapping.[9] Figure 1 shows the MFFC decomposition of a small combinational network.

  A modified MFFC decomposition [CX95] handles circuits with sequential elements, as follows. (1) Given a circuit $H(V, E)$, assign to a set Q all POs and all cells that fanin to sequential cells. (2) Choose an arbitrary cell $v \in Q$, and initialize the MFFC whose output is v, $MFFC_v = \{v\}$. (3) Construct $MFFC_v$ by repeatedly including a transitive fanin u if $OS(u) \subseteq MFFC_v$, and stop when no such cell exists. (4) Let $V = V - MFFC_v$ and update Q by adding all cells whose fanouts include at least one cell in $MFFC_v$. Return to Step (2), stopping when V is empty. Dey, Brglez and Kedem [DBK90a] [DBK90b] earlier proposed *corolla* clustering, which is similar to MFFC decomposition. Another cone-

[9] The intuition is that MFFCs are suited for technology mapping onto lookup table based configurable logic blocks that have a single primary output and a fixed number of primary inputs.

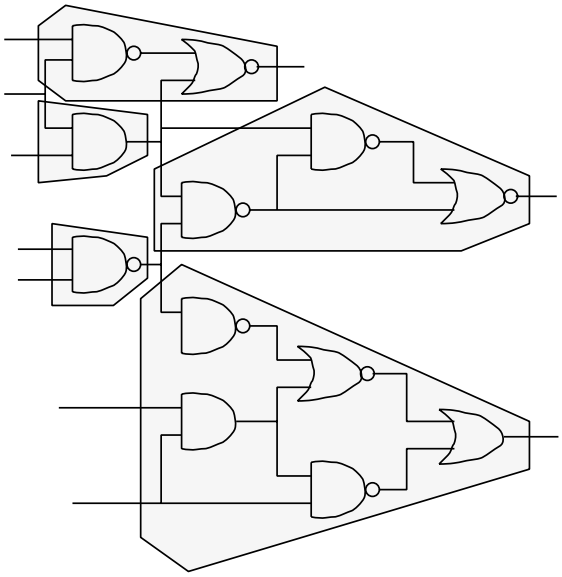

Figure 1: MFFC decomposition of a small network.

based clustering approach is due to Saucier et al. [SBH93].[10] We do not study these methods, since they are at least partly subsumed by the four MFFC clustering variants that we now describe.

- **MFFC (with size bounds).** Our MFFC decomposition implementation is similar to that of [CX95]. The main difficulty with MFFC decomposition is that some MFFCs can be very large; this can be addressed in two ways. (1) We may perform MFFC *splitting* after a large MFFC is formed. Given an MFFC M, let *root*(M) be the root cell of M. To split M, we make *root(M)* into a single cluster. Then, for each cell u in $IS_{root(M)}$, we add $MFFC_u$ to the list of clusters. (2) Alternatively, we may control MFFC size during decomposition, i.e., by modifying Step (3) above so that transitive fanins are included until no such cell exists *or* until the MFFC size limit is reached. Our version of MFFC clustering, as well as the three following variants (MFFCLA, IMFFC, IMFFCLA), adopts the latter approach to enforce size bounds.
- **MFFCLA.** Stricter criteria for a cell to become a member of an MFFC can yield clusters in which cells are more strongly related. Our *Maximum Fanout-free Cone with LookAhead* (MFFCLA) clustering constructs $MFFCLA_v$ such that for any non-PI node w, if $\{OS_u - \{w\}\} \subseteq MFFCLA_v$ for all $u \in IS_w$, then $w \in MFFCLA_v$. A disadvantage of this criterion is that the construction can potentially deadlock when two nodes u and v, both under consideration to be included in an MFFCLA, have a common predecessor w that is not in the MFFCLA.

[10] For a given PO v in a DAG, recall that $Cone_v$ is the set of all $w \in V$ such that there is a directed path from w to v. Saucier et al. form clusters C from the "coarsest common clustering" of the cones of all POs, i.e., C is a cluster if for every cone $Cone_v$, $C \cap Cone_v = C$ or $C \cap Cone_v = \emptyset$. (In partitioning the design into multiple FPGA devices, Saucier et al. define the *cost* of cluster C as $cost(C) = area(C)/|E(C)|$, where $area(C)$ is the total cell area in cluster C. Clusters are merged by iteratively finding C_i with maximum cost and merging it with C_j that maximizes $cost(C_i \cup C_j)$; the merging continues as long as the clusters satisfy prescribed limits on area and IO count.)

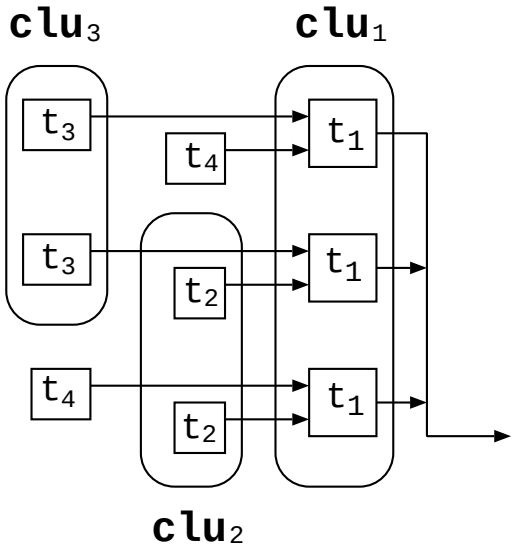

Figure 2: Three clusters formed by datapath regularity extraction. Initially, three cells of type t_1 form cluster clu_1. Three predecessor cells are connected to each of the pin types of the cells in clu_1. Two cells of type t_2 form cluster clu_2, and two cells of type t_3 form cluster clu_3.

  - **IMFFC.** Executing the MFFC clustering heuristic on the netlist with all edge directions reversed yields the *Inverted Maximum Fanout-free Cone (IMFFC)* clustering. This is similar to the corolla-based methods of Dey et al. [DBK90a].
  - **IMFFCLA.** Executing the MFFCLA clustering heuristic on the netlist with all edge directions reversed yields the *Inverted Maximum Fanout-free Cone with LookAhead (IMFFCLA)* clustering.

- **NJ96.** Nijssen and Jess [NJ96] extract "regular" (2-D tilable) structures corresponding to datapath elements from general netlists. This extraction can lead to more regular layouts that are compact and routable, and have better performance. Our implementation, based on [NJ96], has so far focused on "vertical" clustering (i.e., the cells in a cluster correspond to a datapath stage that should be vertically aligned).[11] Essentially, we start with the net containing the greatest number of cells of the same type (i.e., that instantiate the same master library cell). These cells are clustered together. For any given cluster, let its *predecessor* cells be those that fanin to any cell in the cluster. From the current cluster, we cluster together all predecessor cells that instantiate the same library cell. The algorithm continues in this way, traversing the entire circuit until all cells are visited. Figure 2 shows three clusters formed by this approach.

[11] We assume that each bitslice should be placed within a single (horizontal) row, while each datapath stage is vertically aligned perpendicular to the rows. Data flow is parallel to the rows, while control flow is perpendicular to data flow. (A side note: in practice, other information such as bus names can be used in combination with partitioning techniques to aid extraction of datapath regularity. Such is the approach of commercial tools such as Arcadia's Mustang placement preprocessor.)

- **ALG1.** This heuristic involves two steps. (1) The first step performs clustering of sequential cells. Two sequential cells u and v satisfy the *mergeSQ* relation if (a) they instantiate the same master from the cell library, (b) they are connected to a common net, and (c) $IL_u \bigcap IL_v \neq \emptyset$ and $OL_u \bigcap OL_v \neq \emptyset$. Clusters of sequential cells are formed by transitive closure of the *mergeSQ* relation. (2) The second step performs clustering of combinational cells. Two combinational cells u and v satisfy the *mergeCC* relation if $OL_u = OL_v$. Clusters of combinational cells are formed by transitive closure of the *mergeCC* relation.

- **ALG2.** This heuristic also involves two steps. (1) The first step is identical to that of ALG1. (2) In the second step, a combinational cell u is added to a sequential cell cluster if the sequential cell that is closest to u in terms of hypergraph distance ("undirected hop count") is in that cluster. Thus ALG2 clusters sequential and combinational cells together. The total number of clusters is equal to the number of sequential cell clusters in ALG1.

- **ALG3.** As in ALG1 (Step (2)), two combinational cells u and v satisfy the *mergeCC* relation if $OL_u = OL_v$, and clusters of combinational cells are formed by transitive closure of *mergeCC*. However, we do not use the same sequential cell clustering as ALG1. Rather, we put the sequential cells of OL_u into the same cluster as combinational cell u, if the sequential cells have not already been assigned to clusters. Given a cluster area upper bound, we stop clustering when any further clustering violates the bound. The total number of clusters is equal to the number of combinational cell clusters in ALG1.

- **ALG4.**

  Again, two combinational cells u and v satisfy the *mergeCC* relation if $OL_u = OL_v$, and combinational cell clusters are formed by transitive closure of *mergeCC*. These clusters are "tighter" than those of ALG1 and ALG3 since latches are not clustered. Given a cluster area upper bound, we stop when any further clustering violates the bound.

- **ALG5.** Intuitively, cells with low-fanout connections between them should have stronger connectivity than cells with high-fanout connections. Given a cluster area upper bound, ALG5 works as follows. (1) Set $counter = 1$. (2) Find a cell u with $|OS_u| = counter$, if one exists, then cluster u and all cells in OS_u in random order, stopping if any further clustering violates the area bound. Contract the cluster into a single cell, say v, and update the fanout set OS_v. Repeat until no further contraction is possible. (3) Increase *counter* by 1 and repeat Step (2); stop when no two adjacent cells have sum of areas less than the upper bound.

- **ALG6.** ALG6 has the same motivation as ALG5. Given a cluster area upper bound, ALG6 works as follows. (1) Set $counter = 1$. (2) Find a cell u with $|OS_u| = counter$, if one exists, then cluster u and all

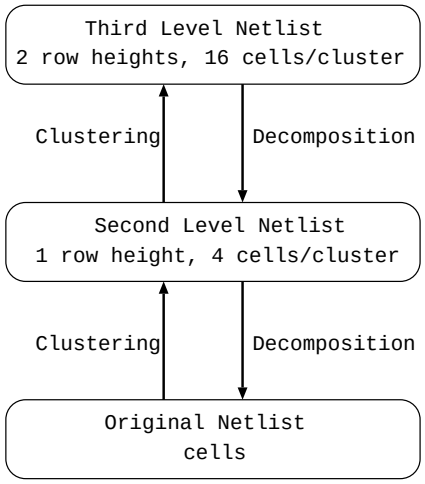

Figure 3: TimberWolf7.0 hierarchical placement

cells in OS_u in random order, stopping if any further clustering violates the area bound. Contract the cluster into a single cell, say v, and update the fanout set OS_v. If any cells are contracted then return to Step (1). (3) Increase *counter* by 1 and repeat Step (2); stop when no two adjacent cells have sum of areas less than the upper bound.

4 Studies of Clustering Effects on Placement

In this section we discuss three previous clustering-based placement approaches, TimberWolf7.0, cone-based placement and MFFC-TW, as well as the Clu-QUAD clustering-based variant of our QUAD placer.

- **TimberWolf7.0**. TimberWolf7.0 [SS95] applies a hierarchical placement methodology (see Figure 3) that combines clustering based on the Absorption objective and simulated annealing (SA). The original netlist is hierarchically clustered into two coarser-grain levels of netlists, then SA is used to place these levels of netlists. Specifically, two clustering phases contract the original netlist into second- and then third-level netlists. Clusters in each higher-level netlist have similar sizes, which greatly aids the annealing placement. The third-level netlist is placed using relatively high temperature, then decomposed back to the second-level netlist, whose cells are initially placed within the boundaries of the higher-level placed cluster. The second-level netlist is then placed using medium temperature, then decomposed back to the original netlist, and low temperature SA is applied.

- **Cone-Based Placement**. Tsay and Lin [TL95] extract cone-based clusters based on structural properties of the circuit, then perform a macro-cell placement treating each cone as a soft macro. The resulting macro-cell placement is then mapped into a row-based placement, and simulated annealing is used to refine the placement result. The cone extraction algorithm starts from the POs or the fan-in

terminals of sequential cells. Breadth-first search is used to traverse the circuit until a sequential cell or a primary input terminal is encountered. The cells that are visited during the search procedure are designated as a single cone and are removed from the circuit. The procedure is repeated until all cells have been clustered. This is followed by macro-cell placement using TimberWolfMC [Sec91] and then the placement is fed to TimberWolf6.0 as its initial placement.

- **MFFC-TW**. Cong and Xu [CX95] use the MFFC decomposition and a "containment tree" heuristic that splits large MFFCs into smaller ones, to obtain clusters with restricted sizes. The clustered circuit is then fed into the TimberWolf6.0 placement package, as follows. (1) Linear placements are found for the cells in each cluster, and the clusters are then contracted into large cells to obtain the clustered circuit. This forces all cells in the same cluster to be placed in the same standard cell row. (2) TimberWolf6.0 is then used to place the clustered circuit and perform global routing. (3) Critical path delay is calculated as a measure of circuit performance.

- **Clustering-Based QUAD**. Our clustering-based QUAD (Clu-QUAD) heuristic is a simple modification of the quadrisection-based QUAD placer [HK97]. Given a clustering solution C, each subnetlist is contracted according to C and a threshold parameter thr before being passed into the partitioner. The threshold thr determines which clusters $clu \in C$ are contracted: if $area(clu) < thr \cdot A$, where A is the total cell area in the subnetlist being partitioned, then cluster clu is contracted to a single cell. Typically, we set thr in the range of $\frac{1}{32}$ to $\frac{1}{64}$. A clustered netlist is formed by this contraction process, then passed into the partitioner. When the partitioning is complete, the clustered netlist is flattened to the original subnetlist, and all cells in the subnetlist are assigned to partitions accordingly. Figure 4 shows the algorithm template for clustering-based QUAD.

5 Experiments and Conclusions

We computed clusterings of the three ACM SIGDA (formerly "MCNC") test cases biomed, avqsmall and avqlarge according to all of the heuristics described in Section 3. If the heuristic admits a cluster area upper bound, we made separate runs with upper bound equal to twice, three times, and five times the area of the largest cell in the netlist. These clusterings are distinguished below by the suffixes -2, -3 and -5. A total of 35 distinct clusterings were obtained for each test case.

We evaluated the clusterings according to seven objectives discussed in Section 2: the C objective, the C' objective, the Density objective with respect to placements obtained from QUAD, SPEED and DOMINO, the Scaled Cost objective, and the Absorption objective. Table 3 lists the desired optimization for each objective, i.e., whether it is to be maximized or minimized. The third column of the table also shows the expected sign of the correlation with the sum of MST costs in the Clu-QUAD placements.

```
Clustering-Based QUAD:
Input: Netlist N with fixed IO locations, a clustering C
Output: cell location cellLoc[]
Variables: N → nRows = number of rows to be placed in netlist N
           N → nCells = number of cells in netlist N
           N → center = the center location of netlist N
           N → level = the hierarchal level of netlist N
1.  newN = removeIOPad(N);
2.  newN → level = 1
3.  Q.push(newN);
4.  currLevel = 0;
5.  while (Q.top() ≠ ∅) do
6.      currLevel = currLevel + 1;
7.      QC.clear(); /* make QC empty */
8.      while (Q.top() → level = currLevel) do
9.          QC.push(Q.pop());
10.     while (true) do
11.         QT ← QC;
12.         while (QT ≠ ∅) do
13.             currN = QT.pop();
14.             cluN = contractNetlist(currN, C);
15.             if (cluN → nRows > 1)
16.                 compute4WayNetVector(cluN);
17.                 (P0, P1, P2, P3) = Quadrisection(cluN);
18.                 k = 3;
19.             else
20.                 compute2WayNetVector(cluN);
21.                 (P0, P1) = Bisection(cluN);
22.                 k = 1;
23.             endif
24.             currN = uncontractNetlist(cluN, C);
25.             for each i = 0 to k do
26.                 /* create subnetlist according to Pi, update netlist region */
27.                 Ni = formSubNetlist(Pi, currN);
28.                 Ni → level = currLevel + 1;
29.                 forall cell ∈ Ni do
30.                     cellLoc[cell] = Ni → center;
31.                 if (Ni → nCells > 1)
32.                     QS.push(Ni);
33.             endfor
34.         endwhile
35.         if (placementCost(cellLoc) < minCost)
36.             minCost = placementCost(cellLoc);
37.             bestCellLoc ← cellLoc;
38.             bestQS ← QS;
39.             QS.clear();
40.         else
41.             cellLoc ← bestCellLoc;
42.             while (bestQS ≠ ∅) do
43.                 Q.push(bestQS.pop());
44.             break;
45.         endif
46.     endwhile
47. endwhile
48. return cellLoc;
```

Figure 4: The clustering-based QUAD algorithm (Clu-QUAD).

Objective	Optimization	Expected Corr w/MST
C	Minimize	Positive
C'	Maximize	Negative
$d-QUAD$	Maximize	Negative
$d-SPEED$	Maximize	Negative
$d-DOMINO$	Maximize	Negative
Scaled Cost	Minimize	Positive
Absorption	Maximize	Negative

Table 3: Expected correlations with total MST cost for different clustering objectives.

Both standard correlation and rank correlation were evaluated between each objective and the Clu-QUAD total MST cost.[12] Table 4 shows the standard correlations between each objective and the Clu-QUAD total

[12] For pairs of quantities $(x_i, y_i), i = 1, \ldots, N$, the linear correlation coefficient r is given by the formula $r = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2}\sqrt{\sum_i (y_i - \bar{y})^2}}$ where $\bar{x}$ is the mean of the x_i's and $\bar{y}$ is the mean of the y_i's. Let R_i be the rank of x_i among the other x's and let S_i be the rank of y_i among the other y's, with ties being assigned appropriate midranks. Then the rank-order correlation coefficient is defined to be the linear correlation coefficient of the ranks.

Objective	Biomed	Avq_Small	Avq_Large
C	-0.150/(-0.258,-0.108)/(-0.352,-0.063)	0.374/(0.264,0.411)/(0.159,0.448)	0.411/(0.317,0.458)/(0.232,0.509)
C'	0.178/(0.144,0.216)/(0.114,0.258)	-0.234/(-0.260,-0.165)/(-0.289,-0.115)	-0.296/(-0.335,-0.242)/(-0.377,-0.185)
$d - QUAD$	-0.032/(-0.108,0.010)/(-0.179,0.046)	-0.280/(-0.305,0.223)/(-0.325,-0.150)	-0.284/(-0.310,-0.225)/(-0.335,-0.139)
$d - SPEED$	-0.039/(-0.122,0.019)/(-0.188,0.064)	-0.289/(-0.323,-0.156)/(-0.347,-0.079)	-0.211/(-0.261,-0.070)/(-0.291,-0.005)
$d - DOMINO$	-0.008/(-0.096,-0.033)/(-0.157,0.066)	-0.236/(-0.267,-0.123)/(-0.295,-0.071)	-0.276/(-0.346,-0.128)/(-0.381,0.003)
Scaled Cost	-0.156/(-0.307,-0.094)/(-0.400,-0.056)	-0.131/(-0.234,-0.047)/(-0.263,0.027)	-0.019/(-0.065,0.107)/(-0.102,0.210)
Absorption	0.012/(-0.033,0.132)/(-0.080,0.183)	0.090/(0.025,0.127)/(-0.079,0.151)	-0.008/(-0.090,0.040)/(-0.185,0.076)
$k \times C'$	-0.009/(-0.095,0.033)/(-0.143,0.097)	-0.333/(-0.353,-0.246)/(-0.372,-0.179)	-0.311/(-0.342,-0.207)/(-0.374,-0.124)

Table 4: Correlation with total MST cost in the Clu-QUAD placement, for different clustering objectives.

MST cost, over the 35 clusterings for each test case. Notice that the C' objective contains a factor of $1/k$, where k is the number of clusters. This factor penalizes clusterings that have a large number of clusters. The remaining part of the objective is a more direct measure of the clustering quality. Indeed, we find that $k \times C'$ has stronger correlation to the Clu-QUAD MST cost than C'.

To ensure that outliers were not corrupting the correlation values, we also computed the best correlations (with the same signs as the original correlations) obtained by removing any single observation, or by removing any two observations. Thus, each entry in Table 4 is a triple of form "Orig / Minus-1 / Minus-2", where Orig is the original value of the correlation between the values of the objective function and the Clu-QUAD MST costs, Minus-1 is the range (min,max) of correlations found with one observation removed, and Minus-2 is the range of correlations found with two observations removed.

We observe that the C objective shows positive correlations, and that the C' and density objectives show negative correlations, as expected. However, both the Scaled Cost and Absorption objectives show some correlations with *opposite* sign (and lower magnitude) than expected. While this is being investigated further, one possible interpretation is that cut-based top-down placers have different clustering requirements than annealing placers. In particular, while max-Absorption clustering strongly benefits TimberWolf, it does not have a noticable effect on Clu-QUAD.

More detailed portraits of the clustering objective evaluations for each test case are given in Tables 5 through 10. CPU (mm:ss) is for Clu-QUAD placement on a 140MHz Sun Ultra-1, using each given clustering. Again, an entry under the "Algorithm" column with suffix -2, -3 or -5 indicates that a cluster area bound of two, three or five times the maximum cell area was applied. The six tables divide into two sets of three: the first set uses Clu-QUAD in default mode, which applies cycling and overlapping metaheuristics to improve the placement quality. The second set of tables turns off the cycling and overlapping, so that Clu-QUAD is more of a "straight" top-down placer. We present both sets of tables in case the cycling and overlapping mask the differences between clustering heuristics.

To further investigate the effectiveness of our clustering heuristics, we also evaluated the placement obtained with a version of TimberWolf. This was an experimental protptype and it was able to take in a single-level clustering file and perform placement accordingly. Table 11 presents the results for Biomed with

the cluster area bound of two, three and five times the maximum cell area. Table 12 presents the results for Avqsmall with a cluster area bound twice the maximum cell area. The second and third columns in the tables list the average and the minimum costs of TimberWolf placements. The entry, CLU-TW, in the two tables, refers to the clustering heuristic used internally by TimberWolf in its stand-alone run. Our results indicate that with TimberWolf, the CLU-TW, ALG5 and ALG6 heuristics perform extremely well. It should, however, be pointed out that the CLU-TW heuristic is hierarchical which might tip the scale a bit in its favour.

In conclusion, we have presented the first analysis of clustering for standard-cell placement in terms of its three basic constituents: the clustering objective, the clustering heuristic, and the benefit to placement. We have identified a new clustering metric that has much stronger correlation to actual placement performance than previous metrics, and we have also developed new clustering heuristics that seem to outperform previous heuristics with respect to *all* known metrics simultaneously.

Our ongoing work is pursuing several directions. First, we are updating the Clu-QUAD placer to accept, and efficiently work with, multilevel clustering hierarchies. At the same time, we are improving our new clustering objectives and heuristics, and exploring multilevel variants. Second, we are performing a second set of experiments with an annealing placer. Of particular interest is whether we find clear evidence that cut-based and annealing placers have different clustering requirements. Finally, recall that traditional placement and clustering formulations rely mostly on netlist topology and ignore many pieces of design information, such as hierarchy naming, bus names, functional information about cells, etc. Following recent works (e.g., [NJ96] [BHB96]) we are exploring methods that incorporate such information into the heuristic clustering.

References

[AHK96] C. J. Alpert, L. W. Hagen, and A. B. Kahng. A hybrid multilevel/ genetic approach for circuit partitioning. In *Proc. ACM/SIGDA Physical Design Workshop*, pages 100–105, 1996.

[AK94] C. J. Alpert and A. B. Kahng. A general framework for vertex orderings, with applications to netlist clustering. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 63–67, 1994.

[AK95] C. J. Alpert and A. B. Kahng. Recent directions in netlist partitioning: A survey. *Integration, the VLSI Journal*, 19:1–81, 1995.

[Alp96] C. J. Alpert. *Multi-way Graph and Hypergraph Partitioning.* PhD thesis, University of California, Los Angeles, 1996.

[BCLS87] T. Bui, S. Chaudhuri, T. Leighton, and M. Sipser. Graph bisection algorithms with good average case behavior. *Combinatorica*, 7(2):171–191, 1987.

[BHB96] D. Behrens, K. Harbich, and E. Barke. Hierarchical partitioning. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 470–477, 1996.

[BHJL89] T. Bui, C. Heigham, C. Jones, and T. Leighton. Improving the performance of the kernighan-lin and simulated annealing graph bisection algorithms. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 775–778, 1989.

[CD93] J. Cong and Y. Ding. On area/depth trade-off in lut-based fpga technology mapping. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 213–218, 1993.

Algorithm	$C(\times 10^8)$	C'	d-QUAD	d-SPEED	d-DOMINO	Scaled Cost	Absorption	MST ($\times 10^6$)	CPU
MFFC-2	42.884	762.907	0.50	0.43	0.46	40.1680	2962.03	3.418	150:31
MFFLA-2	54.358	837.222	0.68	0.59	0.60	48.1355	1356.10	3.362	112:12
IMFFC-2	60.588	491.164	0.63	0.53	0.57	49.0492	760.88	3.393	131:10
IMFFCLA-2	64.416	161.552	0.41	0.38	0.46	49.1261	368.81	3.425	166:36
ALG3-2	38.515	697.814	0.21	0.20	0.27	39.0561	2581.24	3.336	98:33
ALG4-2	38.094	545.263	0.21	0.21	0.27	37.3323	2566.82	3.451	107:29
ALG5-2	28.105	1985.040	0.46	0.40	0.50	26.4842	4482.93	3.417	162:56
ALG6-2	28.066	1988.190	0.46	0.40	0.50	26.4392	4491.41	3.413	74:44
ANNEAL-2	48.695	922.560	0.45	0.45	0.45	35.1237	2948.86	3.413	150:42
BUI-2	46.845	776.706	0.42	0.42	0.42	33.0824	1891.60	3.484	152:22
MFFC-3	42.811	721.314	0.49	0.42	0.44	40.2278	3012.21	3.360	149:27
MFFLA-3	54.358	837.222	0.68	0.59	0.60	48.1355	1356.10	3.362	113:58
IMFFC-3	60.571	492.943	0.63	0.53	0.57	49.3556	757.69	3.365	123:29
IMFFCLA-3	64.316	164.258	0.41	0.38	0.46	49.4178	373.62	3.341	128:19
ALG3-3	36.508	804.109	0.22	0.21	0.29	37.4945	3177.91	3.472	130:55
ALG4-3	36.021	598.969	0.22	0.22	0.29	35.9722	3162.49	3.282	95:28
ALG5-3	29.331	1261.295	0.32	0.29	0.41	26.2365	4371.31	3.437	139:31
ALG6-3	28.450	1255.236	0.32	0.29	0.41	26.2958	4386.09	3.397	69:46
ANNEAL-3	54.368	1162.490	0.34	0.34	0.35	29.3819	3765.15	3.361	179:12
BUI-3	48.154	725.881	0.31	0.31	0.31	28.2191	2459.32	3.530	78:50
MFFC-5	43.113	686.432	0.48	0.41	0.43	40.3409	3029.51	3.429	74:23
MFFLA-5	54.358	837.222	0.68	0.59	0.60	48.1355	1356.10	3.362	64:16
IMFFC-5	60.449	497.584	0.63	0.52	0.47	50.0997	748.47	3.341	81:20
IMFFCLA-5	63.985	170.659	0.41	0.38	0.46	50.1280	384.39	3.359	77:33
ALG3-5	37.651	925.212	0.21	0.21	0.27	39.0401	3465.34	3.434	55:17
ALG4-5	37.007	656.220	0.22	0.21	0.28	36.8310	3447.81	3.355	69:26
ALG5-5	30.991	685.418	0.17	0.15	0.21	20.4178	5176.64	3.415	60:44
ALG6-5	30.953	550.668	0.17	0.15	0.21	20.3724	5212.30	3.276	76:19
ANNEAL-5	60.651	1051.341	0.13	0.13	0.23	23.2743	4166.14	3.325	76:03
BUI-5	57.293	836.787	0.11	0.11	0.22	26.4528	3001.49	3.400	111:13
ALG1	40.597	585.766	0.21	0.21	0.26	37.4777	3765.70	3.369	78:44
ALG2	40.617	598.744	0.15	0.13	0.20	23.3188	4880.04	3.409	83:11
ALG3	41.319	858.095	0.21	0.21	0.25	40.3975	3783.23	3.379	70:25
ALG4	40.597	585.766	0.21	0.21	0.26	37.4777	3765.70	3.369	76:30
NJ	60.020	107.031	0.07	0.09	0.08	42.3464	350.40	3.408	91:38
Correlation	-0.150	0.178	-0.032	-0.039	-0.008	-0.156	0.012		
Rank Corr.	-0.234	0.217	-0.017	-0.011	-0.079	-0.200	0.069		

Table 5: Biomed results for QUAD with Cycling and Overlapping.

[CSZ93] P.K. Chan, M.D.F. Schlag, and J. Zien. Spectral k-way ratio cut partitioning and clustering. In *Symposium on Integrated Systems*, 1993.

[CX95] J. Cong and D. Xu. Exploiting signal flow and logic dependency in standard cell placement. In *Asia and South Pacific Design Automation Conference*, pages 399–404, 1995.

[DBK90a] S. Dey, F. Brglez, and G. Kedem. Corolla based circuit partitioning and resynthesis. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 607–612, 1990.

[DBK90b] S. Dey, F. Brglez, and G. Kedem. Partitioning sequential circuits for logic optimization. In *Proceedings IEEE Intl. Conf. Computer Design*, pages 70–76, 1990.

[DJS94] K. Doll, F. M. Johannes, and G. Sigl. Iterative placement improvement by network flow methods. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(10):1189–1199, 1994.

[GB83] M. K. Goldberg and M. Burstein. Heuristic improvement technique for bisection of vlsi networks. In *Proceedings IEEE Intl. Conf. Computer Design*, pages 122–125, 1983.

[HK92] L. Hagen and A. B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design*, 11(9):1074–1085, 1992.

[HK97] D. J. Huang and A. B. Kahng. New techniques for standard cell placement. In *International Symposium on Physical Design (to appear)*, 1997.

[Hua97] D. J. Huang. *New Techniques for Standard Cell Placement.* PhD thesis, University of California, Los Angeles, 1997.

[KK95] G. Karypis and V. Kumar. Multilevel graph partitioning schemes. In P. Banerjee and P. Boca, editors, *Proceedings of the 1995 International Conference on Parallel Processing*, pages 113–122, 1995.

Algorithm	$C(\times 10^8)$	C'	d-QUAD	d-SPEED	d-DOMINO	Scaled Cost	Absorption	MST ($\times 10^6$)	CPU
MFFC-2	592.285	2222.965	0.46	0.45	0.44	9.9299	8268.22	9.440	292:56
MFFLA-2	707.845	2849.976	0.60	0.59	0.50	10.9978	5147.00	9.547	336:40
IMFFC-2	888.037	240.440	0.40	0.41	0.48	11.6624	581.90	9.803	254:34
IMFFCLA-2	872.858	551.399	0.46	0.41	0.44	11.6172	808.14	9.678	292:17
ALG3-2	345.281	2932.658	0.27	0.25	0.33	8.7114	9320.58	9.755	286:23
ALG4-2	413.114	2569.107	0.26	0.27	0.31	9.0548	7631.99	9.568	290:11
ALG5-2	314.329	6132.865	0.38	0.37	0.44	6.0324	13431.50	9.372	306:10
ALG6-2	309.991	6217.980	0.38	0.37	0.44	5.8788	13644.30	9.578	259:04
ANNEAL-2	669.573	2155.601	0.34	0.35	0.34	8.4687	10305.50	10.755	610:34
BUI-2	389.662	4831.449	0.34	0.34	0.34	7.2276	9889.02	9.768	417:31
MFFC-3	580.196	1800.850	0.44	0.42	0.40	10.1056	8481.36	9.643	333:56
MFFLA-3	716.600	2646.338	0.61	0.59	0.50	11.0334	4957.00	9.614	308:54
IMFFC-3	887.831	240.028	0.40	0.41	0.48	11.6633	586.40	9.574	238:56
IMFFCLA-3	872.308	551.898	0.46	0.41	0.44	11.6196	810.61	9.503	267:22
ALG3-3	332.336	2963.385	0.29	0.25	0.35	8.1952	10941.20	9.521	275:25
ALG4-3	398.490	2475.179	0.28	0.27	0.33	8.6892	9194.62	9.747	292:25
ALG5-3	263.279	4174.701	0.27	0.25	0.31	4.9564	15187.40	9.240	234:10
ALG6-3	261.141	4498.919	0.27	0.25	0.31	4.7761	15354.20	9.650	295:00
ANNEAL-3	802.017	1711.198	0.23	0.23	0.23	7.1982	12354.20	11.461	716:19
BUI-3	317.721	3407.997	0.22	0.23	0.23	5.6817	12907.30	10.158	442:05
MFFC-5	554.331	1517.482	0.42	0.41	0.49	10.3011	9210.72	9.648	285:25
MFFLA-5	691.915	2480.636	0.61	0.61	0.53	11.2244	5645.99	9.602	245:00
IMFFC-5	887.747	240.082	0.40	0.41	0.48	11.6639	587.40	9.624	262:48
IMFFCLA-5	872.028	552.285	0.46	0.41	0.44	11.6216	811.95	9.646	260:51
ALG3-5	319.361	2712.995	0.29	0.25	0.36	7.4558	13985.40	9.607	289:09
ALG4-5	383.892	2253.075	0.28	0.27	0.34	8.1875	12030.40	9.552	233:49
ALG5-5	224.291	4092.341	0.22	0.18	0.25	3.9395	17180.20	9.569	256:51
ALG6-5	217.880	4230.735	0.22	0.18	0.25	3.5712	17394.80	9.663	274:05
ANNEAL-5	917.156	963.022	0.22	0.12	0.22	6.1226	13054.90	11.583	576:10
BUI-5	341.874	2624.259	0.22	0.12	0.22	4.6223	15250.70	9.648	520:56
ALG1	393.529	2266.517	0.30	0.29	0.36	8.1183	13417.70	9.603	346:46
ALG2	313.011	2073.127	0.23	0.17	0.25	5.9702	14901.20	9.519	343:41
ALG3	330.564	2759.434	0.31	0.27	0.39	7.3213	15372.60	9.585	358:51
ALG4	393.529	2266.517	0.30	0.29	0.36	8.1183	13417.70	9.603	344:33
NJ	763.216	611.629	0.09	0.12	0.10	11.4541	435.382	9.836	290:13
Correlation	0.374	-0.234	-0.28	-0.289	-0.236	-0.131	0.090		
Rank Corr.	0.296	-0.213	-0.277	-0.256	-0.209	-0.023	-0.129		

Table 6: Avq_small results for QUAD with Cycling and Overlapping.

[KSJ88] J. M. Kleinhans, G. Sigl, and F. M. Johannes. Gordian: A new global optimization/rectangle dissection method for cell placement. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 506–509, 1988.

[NJ96] R. X. T. Nijssen and J. A. G. Jess. Two-dimensional datapath regularity extraction. In *Proc. ACM/SIGDA Physical Design Workshop*, pages 111–117, 1996.

[Rie95] B. M. Riess. 1995. Personal communication.

[SBH93] G. Saucier, D. Brasen, and J. P. Hiol. Partitioning with cone structures. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 236–239, 1993.

[SDJ91] G. Sigl, K. Doll, and F. M. Johannes. Analytical placement: A linear or a quadratic objective function? In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 427–432, 1991.

[Sec91] C. Sechen. *TimberWolf6.0: Mixed macro/standard cell floor planning, placement and routing package User's Manual.* Yale University, 1991.

[Sou81] J. Soukup. Circuit layout. In *Proc. IEEE*, pages 1281–1304, October 1981.

[SS93] W-J. Sun and C. Sechen. Efficient and effective placements for very large circuits. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 170–177, 1993.

[SS95] W. Swartz and C. Sechen. Timing driven placement for large standard cell circuits. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 211–215, 1995.

[Swa96] W. Swartz, 1996. Personal communication.

[TKH88] R.-S. Tsay, E. S. Kuh, and C.-P. Hsu. PROUD: A sea-of-gates placement algorithm. *IEEE Design & Test of Computers*, 5(6):44–56, 1988.

Algorithm	$C(\times 10^8)$	C'	d-QUAD	d-SPEED	d-DOMINO	Scaled Cost	Absorption	MST $(\times 10^6)$	CPU
MFFC-2	645.508	2670.477	0.48	0.42	0.46	8.3058	10431.90	10.193	361:06
MFFLA-2	806.788	2238.382	0.56	0.50	0.47	9.3050	6068.00	10.424	318:50
IMFFC-2	1026.799	194.552	0.38	0.43	0.48	9.8773	581.90	10.132	356:05
IMFFCLA-2	1009.562	420.873	0.44	0.40	0.44	9.8436	808.14	10.388	321:09
ALG3-2	387.292	3144.574	0.27	0.19	0.32	7.2129	10922.60	10.315	335:39
ALG4-2	449.080	2728.604	0.26	0.21	0.30	7.5561	9255.59	10.456	276:23
ALG5-2	347.967	6384.915	0.39	0.32	0.44	4.9640	15870.70	10.005	488:56
ALG6-2	344.963	6325.556	0.39	0.32	0.44	4.9407	15920.80	10.219	308:45
ANNEAL-2	794.366	2000.846	0.34	0.35	0.35	7.4653	11179.70	11.389	662:48
BUI-2	427.003	5270.577	0.34	0.35	0.35	5.9665	11702.40	10.359	518:23
MFFC-3	608.349	2057.758	0.44	0.37	0.40	8.4260	11310.00	10.403	334:27
MFFLA-3	809.319	2054.596	0.55	0.50	0.47	9.3536	6087.00	10.289	333:34
IMFFC-3	1026.600	194.575	0.38	0.43	0.48	9.8780	586.40	10.201	357:48
IMFFCLA-3	1008.976	421.185	0.44	0.40	0.44	9.8453	810.61	10.137	314:01
ALG3-3	361.947	3088.515	0.29	0.19	0.35	6.7605	12883.60	10.375	332:55
ALG4-3	420.430	2661.598	0.28	0.21	0.32	7.2432	11136.70	10.313	412:47
ALG5-3	281.024	4172.892	0.27	0.20	0.31	4.1252	17831.40	9.908	348:18
ALG6-3	278.816	4459.020	0.27	0.20	0.31	4.1042	17809.30	10.213	264:04
ANNEAL-3	932.901	1741.447	0.23	0.24	0.24	6.2625	13502.50	11.692	945:08
BUI-3	338.947	3801.623	0.22	0.23	0.23	4.7254	14953.40	10.307	528:45
MFFC-5	584.031	1937.967	0.45	0.37	0.40	8.4924	12046.70	10.278	324:21
MFFLA-5	794.565	1997.307	0.57	0.51	0.40	9.4419	6578.99	10.254	285:41
IMFFC-5	1026.518	194.597	0.38	0.43	0.48	9.8784	587.40	10.160	351:52
IMFFCLA-5	1008.540	421.423	0.44	0.40	0.44	9.8466	811.95	10.557	370:27
ALG3-5	345.430	2967.078	0.30	0.18	0.38	6.1967	16193.80	10.142	311:04
ALG4-5	400.803	2450.335	0.29	0.21	0.35	6.8757	14238.90	10.440	322:28
ALG5-5	236.162	3916.524	0.21	0.15	0.25	3.1325	20160.60	10.281	265:18
ALG6-5	232.182	4041.695	0.21	0.15	0.25	2.9688	20245.90	10.327	340:46
ANNEAL-5	1049.803	836.624	0.22	0.22	0.22	5.3474	14192.50	12.331	811:07
BUI-5	354.710	2866.246	0.22	0.22	0.22	3.7941	17555.80	10.355	491:19
ALG1	426.109	2398.296	0.31	0.22	0.38	6.7937	16677.70	10.384	294:41
ALG2	339.820	2098.204	0.22	0.12	0.24	5.0614	17507.80	10.174	310:38
ALG3	386.638	2943.414	0.33	0.19	0.41	6.0239	18633.30	10.120	313:09
ALG4	426.109	2398.296	0.31	0.22	0.38	6.7937	16677.70	10.384	297:19
NJ	863.483	539.477	0.08	0.11	0.09	9.6807	429.18	10.856	407:35
Correlation	0.411	-0.296	-0.284	-0.211	-0.276	-0.019	-0.008		
Rank Corr.	0.292	-0.225	-0.205	-0.127	-0.164	0.092	-0.168		

Table 7: Avq_large results for QUAD with Cycling and Overlapping.

[TL95] Y. Tsai and Y. Lin. A row-based cell placement method that utilizes circuit structure properties. *IEEE Transactions on Computer-Aided Design*, 14(3):393–397, 1995.

[WC91] Y.-C. Wei and C.-K. Cheng. Ratio cut partitioning for hierarchical designs. *IEEE Transactions on Computer-Aided Design*, 10(7):911–921, 1991.

Algorithm	$C(\times 10^8)$	C'	d-QUAD	d-SPEED	d-DOMINO	Scaled Cost	Absorption	MST $(\times 10^6)$	CPU
MFFC-2	42.884	762.907	0.50	0.43	0.46	40.1680	2962.03	3.916	10:37
MFFLA-2	54.358	837.222	0.68	0.59	0.60	48.1355	1356.10	3.652	12:21
IMFFC-2	60.588	491.164	0.63	0.53	0.57	49.0492	760.88	3.824	12:40
IMFFCLA-2	64.416	161.552	0.41	0.38	0.46	49.1261	368.81	3.920	12:17
ALG3-2	38.515	697.814	0.21	0.20	0.27	39.0561	2581.24	3.944	9:02
ALG4-2	38.094	545.263	0.21	0.21	0.27	37.3323	2566.82	3.971	9:55
ALG5-2	28.105	1985.040	0.46	0.40	0.40	26.4842	4482.93	3.824	8:49
ALG6-2	28.066	1988.190	0.46	0.40	0.40	26.4392	4491.41	3.758	8:28
ANNEAL-2	48.695	922.560	0.45	0.45	0.45	35.1237	2948.86	4.301	12:48
BUI-2	46.845	776.706	0.42	0.42	0.42	33.0824	1891.60	4.788	12:44
MFFC-3	42.811	721.314	0.49	0.42	0.44	40.2278	3012.21	3.783	10:43
MFFLA-3	54.358	837.222	0.68	0.59	0.60	48.1355	1356.10	3.652	12:24
IMFFC-3	60.571	492.943	0.63	0.53	0.47	49.3556	757.69	3.808	12:34
IMFFCLA-3	64.316	164.258	0.41	0.38	0.46	49.4178	373.62	3.787	13:04
ALG3-3	36.508	804.109	0.22	0.21	0.29	37.4945	3177.91	3.958	9:51
ALG4-3	36.021	598.969	0.22	0.22	0.29	35.9722	3162.49	3.797	10:31
ALG5-3	29.331	1261.295	0.32	0.29	0.41	26.2365	4371.31	3.871	8:24
ALG6-3	28.450	1255.236	0.32	0.29	0.41	26.2958	4386.09	3.970	8:07
ANNEAL-3	54.368	1162.490	0.34	0.24	0.35	29.3819	3765.15	4.954	12:16
BUI-3	48.154	725.881	0.31	0.21	0.31	28.2191	2459.32	4.805	12:56
MFFC-5	43.113	686.432	0.48	0.41	0.43	40.3409	3029.51	3.738	10:11
MFFLA-5	54.358	837.222	0.68	0.59	0.60	48.1355	1356.10	3.652	12:11
IMFFC-5	60.449	497.584	0.63	0.52	0.57	50.0997	748.47	3.912	11:49
IMFFCLA-5	63.985	170.659	0.41	0.38	0.46	50.1280	384.39	3.939	12:08
ALG3-5	37.651	925.212	0.21	0.21	0.27	39.0401	3465.34	3.962	8:47
ALG4-5	37.007	656.220	0.22	0.21	0.28	36.8310	3447.81	3.799	9:28
ALG5-5	30.991	685.418	0.17	0.15	0.21	20.4178	5176.64	3.960	7:34
ALG6-5	30.953	550.668	0.17	0.15	0.21	20.3724	5212.30	3.929	7:02
ANNEAL-5	60.651	1051.341	0.13	0.13	0.13	23.2743	4166.14	5.265	12:49
BUI-5	57.293	836.787	0.11	0.11	0.12	26.4528	3001.49	4.360	13:09
ALG1	40.597	585.766	0.21	0.21	0.26	37.4777	3765.70	3.867	8:46
ALG2	40.617	598.744	0.15	0.13	0.20	23.3188	4880.04	4.184	7:17
ALG3	41.319	858.095	0.21	0.21	0.25	40.3975	3783.23	3.904	8:06
ALG4	40.597	585.766	0.21	0.21	0.26	37.4777	3765.70	3.867	8:36
NJ	60.020	107.031	0.07	0.09	0.08	42.3464	350.40	4.573	10:06
Correlation	0.253	0.036	-0.409	-0.440	-0.317	-0.433	0.107		
Rank Corr.	0.076	0.058	-0.614	-0.592	-0.481	-0.458	0.127		

Table 8: Biomed results for QUAD without Cycling and Overlapping.

Algorithm	$C(\times 10^8)$	C'	d-QUAD	d-SPEED	d-DOMINO	Scaled Cost	Absorption	MST $(\times 10^6)$	CPU
MFFC-2	592.285	2222.965	0.46	0.45	0.44	9.9299	8268.22	10.861	43:53
MFFLA-2	707.845	2849.976	0.60	0.59	0.50	10.9978	5147.00	10.633	46:59
IMFFC-2	888.037	240.440	0.40	0.41	0.48	11.6624	581.90	10.945	49:16
IMFFCLA-2	872.858	551.399	0.46	0.41	0.44	11.6172	808.14	10.740	49:05
ALG3-2	345.281	2932.658	0.27	0.25	0.33	8.7114	9320.58	11.292	38:32
ALG4-2	413.114	2569.107	0.26	0.27	0.31	9.0548	7631.99	10.971	41:37
ALG5-2	314.329	6132.865	0.38	0.37	0.44	6.0324	13431.50	10.491	37:14
ALG6-2	309.991	6217.980	0.38	0.37	0.44	5.8788	13644.30	10.710	36:15
ANNEAL-2	669.573	2155.601	0.34	0.35	0.44	8.4687	10305.50	20.262	51:06
BUI-2	389.662	4831.449	0.34	0.34	0.44	7.2276	9889.02	12.057	47:45
MFFC-3	580.196	1800.850	0.44	0.42	0.40	10.1056	8481.36	10.576	43:42
MFFLA-3	716.600	2646.338	0.61	0.59	0.50	11.0334	4957.00	10.642	47:18
IMFFC-3	887.831	240.028	0.40	0.41	0.48	11.6633	586.40	10.554	49:57
IMFFCLA-3	872.308	551.898	0.46	0.41	0.44	11.6196	810.61	10.759	48:11
ALG3-3	332.336	2963.385	0.29	0.25	0.35	8.1952	10941.20	10.930	37:52
ALG4-3	398.490	2475.179	0.28	0.27	0.33	8.6892	9194.62	11.101	39:19
ALG5-3	263.279	4174.701	0.27	0.25	0.31	4.9564	15187.40	10.483	34:16
ALG6-3	261.141	4498.919	0.27	0.25	0.31	4.7761	15354.20	10.932	34:24
ANNEAL-3	802.017	1711.198	0.23	0.23	0.23	7.1982	12354.20	26.694	49:40
BUI-3	317.721	3407.997	0.22	0.23	0.23	5.6817	12907.30	12.749	42:10
MFFC-5	554.331	1517.482	0.42	0.41	0.49	10.3011	9210.72	10.687	42:47
MFFLA-5	691.915	2480.636	0.61	0.61	0.53	11.2244	5645.99	10.594	45:46
IMFFC-5	887.747	240.082	0.40	0.41	0.48	11.6639	587.40	10.893	49:20
IMFFCLA-5	872.028	552.285	0.46	0.41	0.44	11.6216	811.95	10.891	48:29
ALG3-5	319.361	2712.995	0.29	0.25	0.36	7.4558	13985.40	11.133	36:30
ALG4-5	383.892	2253.075	0.28	0.27	0.34	8.1875	12030.40	10.804	38:21
ALG5-5	224.291	4092.341	0.22	0.18	0.25	3.9395	17180.20	10.739	32:43
ALG6-5	217.880	4230.735	0.22	0.18	0.25	3.5712	17394.80	10.947	32:15
ANNEAL-5	917.156	963.022	0.22	0.12	0.22	6.1226	13054.90	34.435	47:21
BUI-5	341.874	2624.259	0.22	0.12	0.22	4.6223	15250.70	13.009	42:05
ALG1	393.529	2266.517	0.30	0.29	0.36	8.1183	13417.70	10.813	36:03
ALG2	313.011	2073.127	0.23	0.17	0.25	5.9702	14901.20	11.020	33:44
ALG3	330.564	2759.434	0.31	0.27	0.39	7.3213	15372.60	11.416	35:11
ALG4	393.529	2266.517	0.30	0.29	0.36	8.1183	13417.70	10.813	36:10
NJ	763.216	611.629	0.09	0.12	0.10	11.4541	435.38	11.962	45:04
Correlation	0.340	-0.189	–0.283	-0.331	-0.271	-0.199	0.167		
Rank Corr.	0.006	-0.036	-0.597	-0.625	-0.544	-0.314	0.187		

Table 9: Avq_small results for QUAD without Cycling and Overlapping.

Algorithm	$C(\times 10^8)$	C'	d-QUAD	d-SPEED	d-DOMINO	Scaled Cost	Absorption	MST $(\times 10^6)$	CPU
MFFC-2	645.508	2670.477	0.48	0.42	0.46	8.3058	10431.90	11.528	52:51
MFFLA-2	806.788	2238.382	0.56	0.50	0.47	9.3050	6068.00	11.681	56:22
IMFFC-2	1026.799	194.552	0.38	0.43	0.48	9.8773	581.90	11.420	59:14
IMFFCLA-2	1009.562	420.873	0.44	0.40	0.44	9.8436	808.14	11.525	57:27
ALG3-2	387.292	3144.574	0.27	0.19	0.32	7.2129	10922.60	12.148	47:08
ALG4-2	449.080	2728.604	0.26	0.21	0.30	7.5561	9255.59	12.020	48:21
ALG5-2	347.967	6384.915	0.39	0.32	0.44	4.9640	15870.70	11.610	45:20
ALG6-2	344.963	6325.556	0.39	0.32	0.44	4.9407	15920.80	11.369	46:23
ANNEAL-2	794.366	2000.846	0.34	0.35	0.35	7.4653	11179.70	24.941	60:52
BUI-2	427.003	5270.577	0.34	0.35	0.35	5.9665	11702.40	12.544	55:33
MFFC-3	608.349	2057.758	0.44	0.37	0.40	8.4260	11310.00	11.735	50:15
MFFLA-3	809.319	2054.596	0.55	0.50	0.47	9.3536	6087.00	11.514	54:12
IMFFC-3	1026.600	194.575	0.38	0.43	0.48	9.8780	586.40	11.508	58:24
IMFFCLA-3	1008.976	421.185	0.44	0.40	0.44	9.8453	810.61	11.599	56:02
ALG3-3	361.947	3088.515	0.29	0.19	0.35	6.7605	12883.60	11.882	44:30
ALG4-3	420.430	2661.598	0.28	0.21	0.32	7.2432	11136.70	12.045	46:40
ALG5-3	281.024	4172.892	0.27	0.20	0.31	4.1252	17831.40	11.272	41:47
ALG6-3	278.816	4459.020	0.27	0.20	0.31	4.1042	17809.30	11.372	43:22
ANNEAL-3	932.901	1741.447	0.23	0.24	0.24	6.2625	13502.50	34.463	59:06
BUI-3	338.947	3801.623	0.22	0.23	0.23	4.7254	14953.40	13.373	50:21
MFFC-5	584.031	1937.967	0.45	0.37	0.41	8.4924	12046.70	11.533	50:28
MFFLA-5	794.565	1997.307	0.57	0.51	0.50	9.4419	6578.99	11.515	53:20
IMFFC-5	1026.518	194.597	0.38	0.43	0.48	9.8784	587.40	11.656	57:40
IMFFCLA-5	1008.540	421.423	0.44	0.40	0.44	9.8466	811.95	11.792	57:05
ALG3-5	345.430	2967.078	0.30	0.18	0.38	6.1967	16193.80	12.000	43:09
ALG4-5	400.803	2450.335	0.29	0.21	0.35	6.8757	14238.90	12.349	45:46
ALG5-5	236.162	3916.524	0.21	0.15	0.25	3.1325	20160.60	11.593	40:05
ALG6-5	232.182	4041.695	0.21	0.15	0.25	2.9688	20245.90	11.638	41:02
ANNEAL-5	1049.803	836.624	0.12	0.12	0.12	5.3474	14192.50	43.760	58:11
BUI-5	354.710	2866.246	0.12	0.12	0.12	3.7941	17555.80	13.906	49:19
ALG1	426.109	2398.296	0.31	0.22	0.38	6.7937	16677.70	12.203	45:29
ALG2	339.820	2098.204	0.22	0.12	0.24	5.0614	17507.80	11.625	42:51
ALG3	386.638	2943.414	0.33	0.19	0.41	6.0239	18633.30	11.735	43:19
ALG4	426.109	2398.296	0.31	0.22	0.38	6.7937	16677.70	12.203	44:59
NJ	863.483	539.477	0.08	0.11	0.09	9.6807	429.18	13.032	55:35
Correlation	0.357	-0.204	-0.273	-0.225	-0.247	-0.148	0.109		
Rank Corr.	0.060	-0.012	-0.467	-0.417	-0.361	-0.162	0.089		

Table 10: Avq_large results for QUAD without Cycling and Overlapping.

Algorithm	Avg. MST ($\times 10^6$)	Min. MST ($\times 10^6$)
MFFC-2	3.128	3.085
MFFCLA-2	3.240	3.168
IMFFC-2	3.340	3.246
IMFFCLA-2	3.228	3.088
ALG3-2	5.253	5.102
ALG4-2	3.183	3.102
ALG5-2	3.072	2.979
ALG6-2	3.127	3.050
ANNEAL-2	3.658	3.442
BUI-2	3.272	3.251
MFFC-3	3.123	3.091
MFFCLA-3	3.295	3.258
IMFFC-3	3.298	3.255
IMFFCLA-3	3.216	3.144
ALG3-3	5.338	4.929
ALG4-3	3.218	3.184
ALG5-3	3.308	3.206
ALG6-3	3.252	3.217
ANNEAL-3	3.963	3.689
BUI-3	3.306	3.275
MFFC-5	3.122	3.081
MFFCLA-5	3.274	3.202
IMFFC-5	3.226	3.192
IMFFCLA-5	3.258	3.180
ALG3-5	5.277	4.701
ALG4-5	3.196	3.139
ALG5-5	3.193	3.118
ALG6-5	3.289	3.237
ANNEAL-5	4.091	3.897
BUI-5	3.528	3.334
ALG1	3.382	3.299
ALG2	3.458	3.240
ALG3	5.240	4.777
ALG4	3.327	3.251
NJ	7.070	6.842
CLU-TW	3.087	3.071

Table 11: Biomed results for TimberWolf.

Algorithm	Avg. MST ($\times 10^6$)	Min. MST ($\times 10^6$)
MFFC-2	5.353	5.229
MFFCLA-2	5.313	5.207
IMFFC-2	5.662	5.442
IMFFCLA-2	5.580	5.447
ALG3-2	5.953	5.772
ALG4-2	6.174	5.835
ALG5-2	4.929	4.879
ALG6-2	4.976	4.780
ANNEAL-2	7.422	7.324
BUI-2	5.337	5.313
CLUST1	7.124	6.976
CLUST2	5.333	5.119
CLUST3	7.748	7.594
ALG1	6.125	6.063
ALG2	7.094	6.829
CLU-TW	4.871	4.794

Table 12: Avqsmall results for TimberWolf.