Multi-Objective Hypergraph Partitioning Algorithms for Cut and Maximum Subdomain Degree Minimization *

Navaratnasothie Selvakkumaran and George Karypis
University of Minnesota
Department of Computer Science / Army HPC Research Center
Minneapolis, MN 55455
Technical Report # 03-019

{selva,karypis}@cs.umn.edu

ABSTRACT

In this paper we present a family of multi-objective hypergraph partitioning algorithms based on the multilevel paradigm, which are capable of producing solutions in which both the cut and the maximum subdomain degree are simultaneously minimized. This type of partitionings are critical for existing and emerging applications in VLSI CAD as they allow to both minimize and evenly distribute the interconnects across the physical devices. Our experimental evaluation on the ISPD98 benchmark show that our algorithms produce solutions that when compared against those produced by hMETIS have a maximum subdomain degree that is lower by 5% to 54% while achieving comparable quality in terms of cut.

Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]: Design Aids

General Terms

Algorithms, Experimentation

Keywords

Partitioning, Maximum degree, Placement, Congestion

1. INTRODUCTION

Hypergraph partitioning is an important problem with extensive applications to many areas, including VLSI design [5], efficient storage of large databases on disks [23], information retrieval [28], and data mining [9, 15]. The problem is to partition the vertices of a hypergraph into k equal-size subdomains, such that the number of the hyperedges connecting vertices in different subdomains (called the *cut*) is minimized.

*This work was supported by NSF ACI-0133464, CCR-9972519, EIA-9986042, ACI-9982274, and by Army HPC Research Center contract number DAAD19-01-2-0014.

The importance of the problem has attracted a considerable amount of research interest and over the last thirty years a variety of heuristic algorithms have been developed that offer different cost-quality trade-offs. The survey by Alpert and Kahng [5] provides a detailed description and comparison of various such schemes. Recently a new class of hypergraph partitioning algorithms has been developed [8, 11, 2, 14, 4], that are based upon the multilevel paradigm. In these algorithms, a sequence of successively smaller hypergraphs is constructed. A partitioning of the smallest hypergraph is computed. This partitioning is then successively projected to the next level finer hypergraph, and at each level an iterative refinement algorithm (*e.g.*, KL [18] or FM [10]) is used to further improve its quality. Experiments presented in [2, 14, 4, 26, 3, 6, 17] have shown that multilevel hypergraph partitioning algorithms can produce substantially better solutions than those produced by non-multilevel schemes.

However, despite the success of multilevel algorithms in producing partitionings in which the cut is minimized, this cut is not uniformly distributed across the different subdomains. That is, the number of hyperedges that are being cut by a particular subdomain (referred to as the *subdomain degree*) is significantly higher than that cut by other subdomains. This illustrated in Table 1 that shows the ratios of the maximum subdomain degree over the average subdomain degree of various k-way partitionings obtained for the ISPD98 benchmark [3] using the state-of-the-art hMETIS [16] multilevel hypergraph partitioning algorithm. In many cases, the resulting partitionings contain subdomains whose degree is up to two times higher than the average degree of the remaining subdomains.

For many existing and emerging applications in VLSI CAD, producing partitioning solutions that both minimize the cut and also minimize the maximum subdomain degree is of great importance. For example, EDA tools for existing FPGA devices and emerging deep sub-micron architectures need to both minimize the number of interconnects (which is achieved by minimizing the cut) and also evenly distribute these interconnects across the physical device to eliminate high density interconnect regions (which is achieved by minimizing the maximum subdomain degree).

In this paper we present a family of hypergraph partitioning algorithms based on the multilevel paradigm that are capa-

	4-way	8-way	16-way	32-way	64-way
ibm01	1.27	1.55	1.60	1.70	1.76
ibm02	1.35	1.35	1.43	1.51	1.55
ibm03	1.18	1.43	1.68	1.70	1.84
ibm04	1.28	1.35	1.41	1.72	2.39
ibm05	1.16	1.17	1.24	1.33	1.41
ibm06	1.22	1.46	1.46	1.50	1.63
ibm07	1.29	1.46	1.79	1.94	2.04
ibm08	1.06	1.22	1.45	1.73	2.12
ibm09	1.09	1.23	1.65	1.91	2.31
ibm10	1.23	1.43	1.69	1.78	1.85
ibm11	1.21	1.55	1.54	1.66	2.02
ibm12	1.26	1.47	1.72	2.10	2.15
ibm13	1.31	1.81	1.66	1.91	1.85
ibm14	1.20	1.47	1.46	1.63	1.96
ibm15	1.28	1.51	1.71	1.87	2.09
ibm16	1.22	1.39	1.45	1.70	1.84
ibm17	1.18	1.42	1.52	1.80	2.13
ibm18	1.16	1.61	2.33	2.65	2.78

Table 1: The ratios of the maximum subdomain degree over the average subdomain degree of various solutions for the ISPD98 benchmark.

ble of producing solutions in which both the cut and the maximum subdomain degree are simultaneously minimized. Our algorithms treat the minimization of the maximum subdomain degree as a multi-objective optimization problem that is solved once a high-quality, cut based, k-way partitioning has been obtained. Toward this goal, we present highly effective multi-objective refinement algorithms that are capable to produce solutions that explicitly minimize the maximum subdomain degree and ensure that the cut does not significantly increase.

This approach has a number of inherent advantages. First, by building upon a cut-based k-way partitioning, it leverages the huge body of existing research on this topic, and it can benefit from future improvements. Second, because the initial k-way solution is of extremely high-quality, it allows the algorithm to focus on minimizing the maximum subdomain degree without being overly concerned about the cut of the final solution. Finally, it provides a user-adjustable and predictable framework in which the user can specify how much (if any) deterioration on the cut he or she is willing to tolerate in order to reduce the maximum subdomain degree.

We experimentally evaluated the performance of these algorithms on the ISPD98 [3] benchmark and compared them against the solutions produced by hMETIS [16]. Our experimental results show that our algorithms are capable of producing solutions whose maximum subdomain degree is lower by 5% to 54% while producing comparable solutions in terms of cut. Moreover, the computational complexity of these algorithms is relatively low, requiring on the average no more than twice the amount of time required by hMETIS.

The rest of the paper is organized as follows. Section 2 provides some definitions and describes the notation that is used throughout the paper. Section 3 describes the various aspects of our algorithms. Section 4 experimentally evaluates these algorithms and compares them against hMETIS. Finally, Section 5 provides some concluding remarks and outlines directions of future research.

2. DEFINITIONS AND NOTATION

A *hypergraph* $G = (V, E)$ is a set of vertices V and a set of hyperedges E. Each hyperedge is a subset of the set of vertices V. The *size* of a hyperedge is the cardinality of this subset. A vertex v is said to be *incident* on a hyperedge e, if $v \in e$. Each vertex v and hyperedge e has a weight associated with them and they are denoted by $w(v)$ and $w(e)$, respectively.

A decomposition of V into k disjoint subsets $V_1, V_2, \ldots, V_k$, such that $\bigcup_i V_i = V$ is called a *k-way partitioning* of V. We will use the terms *subdomain* or *partition* to refer to each one of these k sets. A k-way partitioning of V is denoted by a vector P such that $P[i]$ indicates the partition number that vertex i belongs to. We say that a k-way partitioning of V *satisfies a balancing constraint* specified by $[l, u]$, where $l < u$, if for each subdomain V_i, $l \leq \sum_{v \in V_i} w(v) \leq u$. The *cut* of a k-way partitioning of V is equal to the sum of the weights of the hyperedges that contain vertices from different subdomains. The *subdomain degree* of V_i is equal to the sum of the weights of the hyperedges that contain at least one vertex in V_i and one vertex in $V - V_i$. The *maximum subdomain degree* of a k-way partitioning is the highest subdomain degree over all k partitions. The *sum-of-external-degrees* (abbreviated as SOED) of a k-way partitioning is equal to the sum of the subdomain degrees of all the partitions.

Given a k-way partitioning of V and a vertex $v \in V$ that belongs to partition V_i, its *internal degree* denoted by $\mathrm{ID}_i(v)$ is equal to the sum of the weights of its incident hyperedges that contain only vertices from V_i, and its *external degree* with respect to partition V_j denoted by $\mathrm{ED}_j(v)$ is equal to the sum of the weights of its incident hyperedges whose all remaining vertices belong to partition V_j.

The *k-way hypergraph partitioning* problem is defined as follows. Given a hypergraph $G = (V, E)$ and a balancing constraint specified by $[l, u]$, compute a k-way partitioning of V such that it satisfies the balancing constraint and minimizes the cut. The requirement that the size of each partition satisfies the balancing constraint is referred to as the *partitioning constraint*, and the requirement that a certain function is optimized is referred to as the *partitioning objective*.

3. MINIMIZING THE MAXIMUM SUBDOMAIN DEGREE

There are two different approaches for computing a k-way partitioning of a hypergraph. One is based on recursive bisectioning and the other on direct k-way partitioning [13]. In recursive bisectioning, the overall partitioning is obtained by initially bisecting the hypergraph to obtain a two-way partitioning. Then, each of these parts is further bisected to obtain a four-way partitioning, and so on. Assuming that k is a power of two, then the final k-way partitioning can be obtained in $\log(k)$ such steps (or after performing $k - 1$ bisections). In this approach, each partitioning step usually takes into account information from only two partitions, and as such it does not have sufficient information to explicitly minimize the maximum subdomain degree of the resulting k-way partitioning. In principle, additional information can be propagated down at each bisection level

to account for the degrees of the various subdomains. For example, during each bisection step, the change in the degrees of the adjacent subdomains can be taken into account (either explicitly or via variations of terminal-propagation-based techniques [12]) to favor solutions that in addition to minimizing the cut also reduce these subdomain degrees. However, the limitation of such approaches is that they end-up *over-constraining* the problem because not only they try to reduce the maximum subdomain degree of the final k-way partitioning, but they also try to reduce the maximum degree of the intermediate lower-k partitioning solutions.

For this reason, approaches based on direct k-way partitioning are better suited for the problem of minimizing the maximum subdomain degree, as they provide a *concurrent* view of the entire k-way partitioning solution. The ability of direct k-way partitioning to optimize objective functions that depend on knowing how the hyperedges are partitioned across all k partitions has been recognized by various researchers, and a number of different algorithms have been developed to minimize objective functions such as the sum-of-external-degrees, scaled cost, absorption *etc.* [21, 5, 7, 17, 25]). Moreover, direct k-way partitioning can potentially produce much better solutions than a method that computes a k-way partitioning via recursive bisection. In fact, in the context of a certain classes of graphs it was shown that recursive bisectioning can be up to an $O(\log n)$ factor worse than the optimal solution [24].

However, despite the inherent advantage of direct k-way partitioning to naturally model much more complex objectives, and the theoretical results which suggest that it can lead to superior partitioning solutions, a number of studies have shown that existing direct k-way partitioning algorithms for hypergraphs, produce solutions that are in general inferior to those produced via recursive bisectioning [21, 7, 17, 25]. The primary reason for that is the fact that computationally efficient k-way partitioning refinement algorithms are often trapped into local minima, and usually require much more sophisticated and expensive optimizers to climb out of them.

To overcome these conflicting requirements and characteristics, our algorithms for minimizing the maximum subdomain degree combine the best features of the recursive bisectioning and direct k-way partitioning approaches. We achieve this by treating the minimization of the maximum subdomain degree as a post-processing problem to be performed once a high-quality k-way partitioning has been obtained. Specifically, we use existing state-of-the-art multilevel-based techniques [14, 16] to obtain an initial k-way solution via repeated bisectioning, and then refine this solution using various k-way partitioning refinement algorithms that (i) explicitly minimize the maximum subdomain degree, (ii) ensure that the cut does not significantly increase, and (iii) ensure that the balancing constraints of the resulting k-way partitioning are satisfied.

This approach has a number of inherent advantages. First, by building upon a cut-based k-way partitioning, it leverages the huge body of existing research on this topic, and it can benefit from future improvements. Second, in terms of cut, its initial k-way solution is of extremely high-quality, thus allowing us to primarily focus on minimizing the maximum subdomain degree without being overly concerned about the cut of the final solution (as long as the partitioning is not significantly perturbed). Third, it allows for a user-adjustable and predictable framework in which the user can specify how much (if any) deterioration on the cut he or she is willing to tolerate in order to reduce the maximum subdomain degree.

To actually perform the maximum subdomain-degree focused k-way refinement we developed two classes of algorithms. Both of them treat the problem as a multi-objective optimization problem but they differ on the starting point of that refinement. Details on the exact multi-objective formulation and the refinement algorithms are provided in the rest of this section.

3.1 Multi-Objective Formulation

In general, the objectives of producing a k-way partitioning that both minimizes the cut and the maximum subdomain degree are reasonably well correlated with each other, as partitionings with low cuts will also tend to have low maximum subdomain degrees. However, this correlation is not perfect, and these two objectives can actually be at odds with each other. That is, a reduction in the maximum subdomain degree may only be achieved if the cut of the partitioning is increased. This situation arises with vertices that are adjacent to vertices that belong to more than two subdomains. For example, consider a vertex v that belongs to the maximum degree partition V_i and let V_q and V_r be two other partitions such that v is connected to vertices in V_i, V_q, and V_r. Now, if $\mathrm{ED}_q(v) - \mathrm{ID}_i(v) < 0$ and $\mathrm{ED}_r(v) - \mathrm{ID}_i(v) < 0$, then the move of v to either partitions V_q or V_r will increase the cut but if $\mathrm{ED}_q(v) + \mathrm{ED}_r(v) - \mathrm{ID}_i(v) > 0$, then moving v to either V_q or V_r will actually decrease V_i's subdomain degree. Thus, in order to develop effective algorithms that explicitly minimize the maximum subdomain degree and the cut, these two objectives need to be coupled together into a multi-objective framework that allows the optimization algorithm to intelligently select the preferred solution.

The problem of multi-objective optimization within the context of graph and hypergraph partitioning has been extensively studied in the literature [22, 1, 27, 20, 19] and two general approaches have been developed for combining multiple objectives. The first approach keeps the different objectives separate and couples them by assigning to them different priorities. Essentially in this scheme, a solution that optimizes the highest priority objective the most is always preferred and the lower priority objectives are used as *tie-breakers* (*i.e.*, used to select among equivalent solutions in terms of the higher priority objectives). The second approach creates an explicit multi-objective function that numerically combines the individual functions. For example, a multi-objective function can be obtained as the weighted sum of the individual objective functions. In this scheme, the choice of the weight values is used to determine the relative importance of the various objectives. One of the advantages of such an approach is that it tends to produce somewhat more natural and predictable solutions as it will prefer solutions that to certain extent, optimize all different objective functions.

In our algorithms we used both of these methods to combine the two different objectives. Specifically, our priority-

based scheme produces a multi-objective solution in which the maximum subdomain degree is the highest priority objective and the cut is the second highest. This choice of priorities was motivated by the fact that within our framework, the solution is already at a local minima in terms of cut; thus, focusing on the maximum subdomain degree is a natural choice. Our combining multi-objective function couples the different objectives using the following formula

$$\text{Cost} = \alpha(\text{MaximumDegree}) + \beta(\text{Cut}), \qquad (1)$$

where *MaximumDegree* is the maximum subdomain degree, *Cut* is the hyperedge cut, and α and β are two user-specified weights indicating the relative importance of these objectives.

In addition, in both of these schemes, we break ties in favor of solutions that lead to lower sum-of-external-degrees. This was motivated by the fact that lower SOED solutions may lead to subsequent improvements in either one of the main objective functions. Also, if a gain of the move is tied even after considering SOED, the ability of the move to improve area balancing is considered for tie breaking.

3.2 Direct Multi-Phase Refinement

Our first k-way refinement algorithm for the multi-objective problem formulations described in Section 3.1 is based on the *multi-phase refinement* approach implemented by hMETIS and was initially described in [14]. The idea behind multi-phase refinement is quite simple. It consists of two phases, namely a coarsening and an uncoarsening phase. The uncoarsening phase is identical to the uncoarsening phase of the multilevel hypergraph partitioning algorithm [14]. The coarsening phase, called *restricted coarsening* [14], however is somewhat different, as it preserves the initial partitioning that is input to the algorithm. Given a hypergraph G and a partitioning P, during the coarsening phase a sequence of successively coarser hypergraphs and their partitionings is constructed. Let (G_i, P_i) for $i = 1, 2, \ldots, m$, be the sequence of hypergraphs and partitionings. Given a hypergraph G_i and its partitioning P_i, restricted coarsening will collapse vertices together that belong to only one of the two partitions. The partitioning P_{i+1} of the next level coarser hypergraph G_{i+1} is computed by simply inheriting the partition from G_i. By constructing G_{i+1} and P_{i+1} in this way we ensure that the number of hyperedges cut by the partitioning is identical to the number of hyperedges cut by P_i in G_i. The set of vertices to be collapsed together in this restricted coarsening scheme can be selected by using any of the coarsening schemes that have been previously developed [14]. In our algorithm, we use the first-choice scheme described in [17], as it leads to the best overall solutions [16].

Due to the randomization in the coarsening phase, successive runs of the multi-phase refinement algorithm can lead to additional improvements of the partitioning solution. For this reason, in our algorithm we perform multiple such iterations and the entire process is stopped when the solution quality does not improve in successive iterations. Such an approach is identical to the V-cycle refinement algorithm used by hMETIS [16].

The actual k-way partitioning refinement at a given level during the uncoarsening phase is performed using a greedy algorithm that is motivated by a similar algorithm using in the direct k-way partitioning algorithm of hMETIS. More precisely, the greedy k-way refinement algorithm works as follows. Consider a hypergraph $G = (V, E)$, and its partitioning vector P. The vertices are visited in a random order. Let v be such a vertex, let $P[v] = a$ be the partition that v belongs to. If v is a node internal to partition a then v is not moved. If v is at the boundary of the partition, then v can potentially be moved to one of the partitions $N(v)$ that vertices adjacent to v belong to (the set $N(v)$ is often refer to as the *neighborhood* of v). Let $N'(v)$ be the subset of $N(v)$ that contains all partitions b such that movement of vertex v to partition b does not violate the balancing constraint. Now the partition $b \in N'(v)$ that leads to the greatest positive reduction in the multi-objective function is selected and v is moved to that partition.

3.3 Aggressive Multi-Phase Refinement

One of the potential problems with the multi-objective refinement algorithm described in Section 3.2 is that it is limited in the extent to which it can make large-scale perturbations on the initial k-way partitioning produced by the cut-focused recursive-bisectioning algorithm. This is due to the combination of two factors. First, the greedy, non-hill climbing nature of its refinement algorithm limits the perturbations that are explored, and second, since it is based on an FM-derived framework, it is constrained to make moves that do not violate the balancing constraints of the resulting solution. As a result (shown later in in our experiments (Section 4)), it tends to produce solutions that retain the low-cut characteristics of the initial k-way solution, but it does not significantly reduce the maximum subdomain degree. Ideally, we will like a multi-objective refinement algorithm that is capable of effectively *exploring* the entire space of possible solutions in order to select the one that best optimizes the particular multi-objective function.

Toward this goal, we developed a multi-objective refinement algorithm that allows large-scale perturbations of the partitioning produced by the recursive bisectioning algorithm. This algorithm consists of five major steps as follows. Given the initial k-way partitioning, in the first step, the algorithm proceeds to further subdivide each of these partitions into 2^l parts (where l is a user specified parameter). During the second step, this $2^l k$-way partitioning is refined using the direct multi-phase refinement algorithm described in Section 3.2 to optimize the particular multi-objective function. Each of the resulting $2^l k$ partitions are then collapsed into single nodes, that we will refer to them as *macro nodes*. Now, during the third step, a k-way partitioning of these macro nodes is computed, such that each partition has exactly 2^l macro nodes. In the fourth step, the quality in terms of the particular multi-objective function of the resulting macro-node level partitioning is improved using a randomized pair-wise node *swapping* algorithm. In this algorithm, two nodes belonging to different partitions are randomly selected and the quality of the partitioning resulting by their swap is evaluated in terms of the particular multi-objective function. If that swap leads to a better solution, the swap is performed, otherwise it is not. Finally, in the fifth step, the macro-node based partitioning is used to induce a partitioning of the original hypergraph, which is then further improved using the direct multi-phase refinement algorithm described in

Section 3.2.

The key idea in the above algorithm is the macro-node-level swapping-based refinement algorithm. This algorithm allows us to move large portions of the hypergraph between partitions without having to either violate the balancing constraints or rely on a sequence of small vertex-moves inorder to achieve the same effect. Moreover, because by construction, each macro-node corresponds to a good cluster (as opposed to a random collection of nodes) of roughly the same size, such swaps can indeed lead to improved quality. Note that the choice of the randomized swapping-based refinement approach was primarily done because of its low computational complexity, and in principle, Kernighan-Lin-based direct k-way refinement algorithms can be used instead.

One of the key elements of this *aggressive* refinement algorithm is the method used to obtain the initial k-way partitioning of the macro-nodes. In our study we implemented two different approaches for computing that partitioning. The first approach focuses on computing an initial partitioning that has low cut, by *inheriting* the original k-way partitioning of the hypergraph. We will refer to this as the *Cut-Focused Macro-Node Partitioning* approach. On the other hand, the second approach focuses on computing an initial partitioning that has low maximum subdomain degree by greedily combining macro-nodes that lead to the smallest maximum subdomain degree. For $l = 1$, this combining is done by sorting all possible pairings of macro-nodes in increasing order of their resulting subdomain degree, and then traversing the list in that order to identify the pairs of *unmatched* macro-nodes to form the initial partitioning. When $l > 1$, such an approach is not computationally feasible and for this reason we repeatedly apply the above scheme l times. We will refer to this as the *Max-Degree-Focused Macro-Node Partitioning* approach.

Finally, the key parameter of this scheme is the value of l, which controls the granularity of the macro-nodes that are used. In particular, the effectiveness of the randomized swapping-based refinement can be affected both for small as well as large values of l. Small values may lead to large macro-nodes whose swaps do not improve the quality, whereas large values may lead to small macro-nodes that require a *coordinated* sequence of swaps (which are not performed by our greedy algorithm) to achieve the desired perturbations. Moreover, large values of l have the additional drawback of increasing the overall runtime of the algorithm as it requires more time to obtain the initial clusters and more refinement time. Fortunately, the fact that the resulting solution is refined at the end using the direct multi-phase refinement algorithm allows this approach to use reasonably small values of l and still achieve good results (as the experiments in Section 4 show) because the final multi-phase refinement step is capable of performing the type of perturbations that will be performed for large values of l. In particular, our experiments show that $l = 2$ leads to the best overall results.

4. EXPERIMENTAL RESULTS

We experimentally evaluated our multi-objective partitioning algorithms on the 18 hypergraphs that are part of the ISPD98 circuit partitioning benchmark suite [3]. The characteristics of these hypergraphs are shown in Table 2. For each of these circuits, we computed a 4-, 8-, 16-, 32-, and 64-way partitioning solution using the recursive bisection-based partitioning routine of hMETIS 1.5.3 [16] and the various algorithms that we developed for minimizing the maximum subdomain degree. The hMETIS solutions were obtained by using a 49–51 bisection balance constraint and hMETIS's default set of parameters. Since these balance constraints are specified at each bisection level, the final k-way partitioning may have a somewhat higher load imbalance. To ensure that the results produced by our algorithm can be easily compared against those produced by hMETIS, we used the resulting minimum and maximum partition sizes obtained by hMETIS as the balancing constraints for our multi-objective k-way refinement algorithm.

Benchmark	No. of vertices	No. of hyperedges
ibm01	12506	14111
ibm02	19342	19584
ibm03	22853	27401
ibm04	27220	31970
ibm05	28146	28446
ibm06	32332	34826
ibm07	45639	48117
ibm08	51023	50513
ibm09	53110	60902
ibm10	68685	75196
ibm11	70152	81454
ibm12	70439	77240
ibm13	83709	99666
ibm14	147088	152772
ibm15	161187	186608
ibm16	182980	190048
ibm17	184752	189581
ibm18	210341	201920

Table 2: The characteristics of the hypergraphs used to evaluate our algorithm.

The quality of the solutions produced by our algorithm and those produced by hMETIS were evaluated by looking at three different quality measures, which are the maximum subdomain degree, the cut, and the average subdomain degree. To ensure the statistical significance of our experimental results, these measures were averaged over ten different runs for each particular set of experiments.

Furthermore, due to space constraints, our comparisons against hMETIS are presented in a summary form, which shows the relative maximum subdomain degree ($RMax$), relative cut ($RCut$), and relative average degree ($RDeg$) achieved by our algorithms over those achieved by hMETIS averaged over the entire set of 18 benchmarks. To ensure the meaningful averaging of these ratios, we first took their $\log_2$-values, calculated their mean μ, and then used 2^{μ} as their average. This method ensures that ratios corresponding to comparable degradations or improvements (*i.e.*, ratios that are less than or greater than one) are given equal importance.

4.1 Direct Multi-Phase Refinement

Our first set of experiments was focused on evaluating the effectiveness of the direct multi-phase refinement algorithm described in Section 3.2. Toward this goal we performed a series of experiments using both formulations of the multi-

objective problem definition described in Section 3.1. The performance achieved in these experiments relative to those obtained by hMETIS's recursive bisectioning algorithm is shown in Table 3. Specifically, this table shows four sets of results. The first set uses the priority-based multi-objective formulation whereas the remaining three sets use Equation 1 to combine the two different objectives. The objectives were combined using three different values of α, namely $1, 2$, and k (where k is the number of partitions that is computed), and β was kept fixed at one.

The results of Table 3 show that irrespective of the number of partitions or the particular multi-objective formulation, the direct multi-phase refinement algorithm produces solutions whose average quality along each one of the three different quality measures is better than the corresponding solutions produced by hMETIS. As expected, the relative improvements are higher for the maximum subdomain degree. In particular, depending on the number of partitions, the direct multi-phase refinement algorithm reduces the maximum subdomain degree by 5% to 17%. The relative improvements increase as the number of partitions increase, because as the results in Table 1 showed, these are the partitioning solutions in which the maximum subdomain degree is significantly higher than the average and thus there is significantly more room for improvement.

Furthermore, the direct multi-phase refinement algorithm also leads to partitionings that on the average have lower cut and average subdomain degree. Specifically, the cut tends to improve by 1% to 4%, whereas the average subdomain degree improves by 5% to 14%. Finally, comparing the different multi-objective formulations we can see that in general, there are very few differences between them, with both of them leading to comparable solutions.

4.2 Aggressive Multi-Phase Refinement

Our second set of experiments was focused on evaluating the effectiveness of the aggressive multi-phase refinement algorithm described in Section 3.3. Toward this goal we performed a series of experiments in which we used both formulations of the multi-objective problem definition, different values of l, and both methods for computing the initial macro-node level-based partitioning. The performance achieved in these experiments relative to those obtained by hMETIS's recursive bisectioning algorithm is shown in Table 4. Specifically, for each value of l, this table shows four sets of results. The first two sets were obtained using the priority-based multi-objective formulation whereas the remaining two sets used the combining scheme. Due to space constraints, we only present results in which the two objectives were combined using $\alpha = k$, and $\beta = 1$. Finally, for each set of experiments, Table 4 shows the results obtained for the cases in which the initial macro-node partitioning was computed using the cut- and the max-degree-focused approaches.

From these results, we can observe a number of general trends about the performance of the aggressive multi-phase refinement algorithm and its sensitivity to the various parameters. In particular, as l increases from one to two (*i.e.*, each partition is further subdivided into two or four parts), the effectiveness of the multi-objective partitioning algorithm to produce solutions that have lower maximum subdomain degree compared to the solutions obtained by hMETIS, improves. In general, for $l = 1$, the multi-objective algorithm reduces the maximum subdomain degree by 5% to 38%, whereas for $l = 2$, the corresponding improvements range from 7% to 53%. However, these improvements lead to solutions in which the cut and the average subdomain degree obtained for $l = 2$ are somewhat higher than those obtained for $l = 1$. For example, for $l = 1$, the multi-objective algorithm is capable of improving the cut over hMETIS by 0% to 3%, whereas for $l = 2$, the multi-objective algorithm leads to solutions whose cut is up to 5% worse than those obtained by hMETIS. Note that these observations are to a large extent independent of the particular multi-objective formulation or the method used to obtain the initial macro-node-level partitioning.

For $l = 3$, the trend of continuing improvements in the maximum subdomain degree does not hold, and in general, the multi-objective algorithm leads to solutions that are worse than those obtained for $l = 2$. We believe that the reason for that is the fact that, as discussed in Section 3.3, at this level of granularity, the macro-node level swapping scheme is not very effective because it operates on relatively small macro-nodes and requires coordinated exchange of nodes in order to be effective.

Also, these results show that the aggressive multi-phase refinement algorithm is to a large extent insensitive on the particular multi-objective function and scheme used to compute the initial macro-node partitioning. The only exception occurs for the cut-focused initial partitioning scheme, for which the priority-based scheme leads to consistently better solutions that those obtained by the combined scheme. The cause of this performance difference is currently under investigation.

Finally, comparing the results obtained by the aggressive multi-phase refinement with the corresponding results obtained by the direct multi-phase refinement algorithm (Tables 4 and 3), we can see that in terms of the maximum subdomain degree, the aggressive scheme leads to substantially better solutions than those obtained by the direct scheme, whereas in terms of the cut and the average subdomain degree, the direct scheme is superior. These results are in agreement with the design principles behind these two multi-phase refinement schemes for the multi-objective optimization problem at hand, and illustrate that the former is capable of making relatively large perturbations on the initial partitioning obtained by recursive bisectioning, as long as these perturbations improve the multi-objective function. In general, the aggressive multi-phase refinement scheme with $l = 1$, dominates the direct scheme, as it leads to better improvements in terms of maximum subdomain degree and still improves over hMETIS in terms of cut and average degree. However, if the goal is to achieve the highest reduction in the maximum average degree, then the aggressive scheme with $l = 2$ should be the preferred choice, as it does so with relatively little degradation on the cut.

4.3 Runtime Complexity

Table 5 shows the amount of time required by the various multi-objective partitioning algorithms using either direct

	Prioritized			Combined, $\alpha = 1, \beta = 1$			Combined, $\alpha = 2, \beta = 1$			Combined, $\alpha = k, \beta = 1$		
k	RMax	RCut	RDeg	RMax	RCut	RDeg	RMax	RCut	RDeg	RMax	RCut	RDeg
4	0.955	0.981	0.948	0.940	0.967	0.934	0.928	0.964	0.931	0.929	0.967	0.934
8	0.890	0.967	0.913	0.877	0.947	0.892	0.886	0.952	0.897	0.881	0.959	0.906
16	0.884	0.969	0.898	0.876	0.958	0.886	0.886	0.965	0.894	0.886	0.966	0.894
32	0.865	0.967	0.886	0.874	0.959	0.874	0.871	0.963	0.877	0.870	0.964	0.878
64	0.851	0.970	0.880	0.864	0.966	0.872	0.876	0.970	0.875	0.859	0.969	0.875

Table 3: Direct Multi-Phase Refinement Results. *RMax*, *RCut*, and *RDeg* are the average maximum subdomain degree, cut, and average subdomain degree, respectively of the multi-objective solution relative to hMETIS. Numbers less than one indicate that the multi-objective algorithm produces solutions that have lower maximum subdomain degree, cut, or average subdomain degree than those produced by hMETIS.

$l = 1$												
	Prioritized						Combined, $\alpha = k, \beta = 1$					
	Cut-Focused			Max-Degree-Focused			Cut-Focused			Max-Degree-Focused		
k	RMax	RCut	RDeg	RMax	RCut	RDeg	RMax	RCut	RDeg	RMax	RCut	RDeg
4	0.923	0.989	0.955	0.927	0.990	0.958	0.910	0.970	0.939	0.904	0.972	0.941
8	0.842	0.984	0.934	0.838	0.995	0.945	0.832	0.974	0.923	0.834	0.992	0.943
16	0.799	0.994	0.932	0.787	1.005	0.942	0.813	0.994	0.929	0.795	1.000	0.935
32	0.757	0.991	0.919	0.754	0.993	0.923	0.797	0.992	0.919	0.758	0.991	0.917
64	0.722	0.993	0.911	0.724	0.996	0.916	0.758	0.992	0.903	0.721	0.993	0.905

$l = 2$												
	Prioritized						Combined, $\alpha = k, \beta = 1$					
	Cut-Focused			Max-Degree-Focused			Cut-Focused			Max-Degree-Focused		
k	RMax	RCut	RDeg	RMax	RCut	RDeg	RMax	RCut	RDeg	RMax	RCut	RDeg
4	0.932	0.999	0.966	0.938	1.021	0.991	0.902	0.974	0.943	0.905	0.992	0.963
8	0.824	1.011	0.963	0.825	1.046	1.004	0.821	0.994	0.945	0.814	1.041	1.001
16	0.760	1.020	0.971	0.749	1.049	1.008	0.786	1.014	0.962	0.751	1.048	1.003
32	0.702	1.021	0.969	0.693	1.041	0.991	0.741	1.019	0.958	0.689	1.033	0.976
64	0.663	1.028	0.971	0.654	1.040	0.983	0.718	1.032	0.963	0.652	1.041	0.974

$l = 3$												
	Prioritized						Combined, $\alpha = k, \beta = 1$					
	Cut-Focused			Max-Degree-Focused			Cut-Focused			Max-Degree-Focused		
k	RMax	RCut	RDeg	RMax	RCut	RDeg	RMax	RCut	RDeg	RMax	RCut	RDeg
4	0.958	1.011	0.977	1.007	1.121	1.091	0.911	0.976	0.943	0.950	1.058	1.029
8	0.847	1.006	0.957	0.848	1.119	1.088	0.834	0.988	0.937	0.842	1.109	1.073
16	0.768	1.018	0.964	0.759	1.101	1.070	0.791	1.012	0.952	0.754	1.077	1.034
32	0.720	1.020	0.968	0.697	1.095	1.059	0.759	1.023	0.964	0.700	1.064	1.010
64	0.727	1.035	0.977	0.701	1.100	1.052	0.788	1.050	0.980	0.663	1.066	1.006

Table 4: Aggressive Multi-Phase Refinement Results. *RMax*, *RCut*, and *RDeg* are the average maximum subdomain degree, cut, and average subdomain degree, respectively of the multi-objective solution relative to hMETIS. Numbers less than one indicate that the multi-objective algorithm produces solutions that have lower maximum subdomain degree, cut, or average subdomain degree than those produced by hMETIS.

or aggressive multi-phase refinement. For each value of k and particular multi-objective algorithm, this table shows the total amount of time that was required to partition all 18 benchmarks relative to the amount of time required by hMETIS to compute the corresponding partitionings. From these results we can see that the multi-objective algorithm that uses the direct multi-phase refinement is the least computationally expensive and requires around 50% more time than hMETIS does. On the other hand, the time required by the aggressive multi-phase refinement schemes is somewhat higher and increases with the value of l. However, even for this algorithm, its overall computational requirements are relatively small. For instance, for $l = 1$ and $l = 2$ (the cases in which the aggressive multi-phase refinement scheme led to the best results) it only requires two and three times more time than hMETIS, respectively.

k	Direct	Aggres., $l = 1$	Aggres., $l = 2$	Aggres., $l = 3$
4	1.431	2.081	2.794	3.809
8	1.399	2.151	2.990	3.924
16	1.397	2.029	3.018	3.584
32	1.450	2.018	2.763	3.599
64	1.535	2.060	3.067	4.522

Table 5: The amount of time required by the multi-objective algorithms relative to that required by hMETIS.

5. CONCLUSIONS AND FUTURE WORK

In this paper we presented a family of multi-objective hypergraph partitioning algorithms for computing k-way partitionings that simultaneously minimize the cut and the maximum subdomain degree of the resulting partitions. Our experimental evaluation showed that these algorithms are quite effective in optimizing these two objectives with relatively low computational requirements. The key factor contributing to the success of these algorithms was the idea of

focusing on the maximum subdomain degree objective once a good solution with respect to the cut has been identified. We believe that such a framework can be applied to a number of other multi-objective problems involving objectives that are reasonably well-correlated with each other.

The multi-objective algorithms presented here can be improved in a number of directions. In particular, our results showed that the aggressive multi-phase refinement approach, though promising, can lead to worse solutions for relatively large values of l. Using and developing better and more powerful refinement algorithms at the macro-node level can potentially address some of these shortcomings. Also, our work so far was focused on producing multi-objective solutions, which satisfy the same balancing constraints as those resulting from the initial recursive bisectioning based solution. However, additional improvements can be obtained by relaxing the lower-bound constraint. Our preliminary results with such an approach appears promising.

6. REFERENCES

[1] C. Ababei, N. Selvakkumaran, K. Bazargan, and G. Karypis. Multi-objectivecircuit partitioning for cutsize and path-based delay minimization. In *Proceedings of ICCAD*, 2002. Also available on WWW at URL http://www.cs.umn.edu/~karypis.

[2] C. Alpert and A. Kahng. A hybrid multilevel/genetic approach for circuit partitioning. In *Proceedings of the Fifth ACM/SIGDA Physical Design Workshop*, pages 100–105, 1996.

[3] C. J. Alpert. The ISPD98 circuit benchmark suite. In *Proc. of the Intl. Symposium of Physical Design*, pages 80–85, 1998.

[4] C. J. Alpert, J. H. Huang, and A. B. Kahng. Multilevel circuit partitioning. In *Proc. of the 34th ACM/IEEE Design Automation Conference*, 1997.

[5] C. J. Alpert and A. B. Kahng. Recent directions in netlist partitioning. *Integration, the VLSI Journal*, 19(1-2):1–81, 1995.

[6] A. E. Caldwell, A. B. Kahng, and I. L. Markov. Improved algorithms for hypergraph bipartitioning. In *Asia and South Pacific Design Automation Conference*, pages 661–666, 2000.

[7] J. Cong and S. K. Lim. Multiway partitioning with pairwise movement. In *Proceedings of ICCAD*, pages 512–516, 1998.

[8] J. Cong and M. L. Smith. A parallel bottom-up clustering algorithm with applications to circuit partitioning in vlsi design. In *Proc. ACM/IEEE Design Automation Conference*, pages 755–760, 1993.

[9] R. Cooley, B. Mobasher, and J. Srivastava. Web mining: Information and pattern discovery on the world wide web. In *International Conference on Tools with Artificial Intelligence*, pages 558–567, Newport Beach, 1997. IEEE.

[10] C. M. Fiduccia and R. M. Mattheyses. A linear time heuristic for improving network partitions. In *In Proc. 19th IEEE Design Automation Conference*, pages 175–181, 1982.

[11] S. Hauck and G. Borriello. An evaluation of bipartitioning technique. In *Proc. Chapel Hill Conference on Advanced Research in VLSI*, 1995.

[12] B. Hendrickson, R. Leland, and R. V. Driessche. Enhancing data locality by using terminal propagation. In *Proceedings of the 29th Hawaii International Conference on System Science*, 1996.

[13] G. Karypis. Multilevel hypergraph partitioning. In J. Cong and J. Shinnerl, editors, *Multilevel Optimization Methods for VLSI*, chapter 6. Kluwer Academic Publishers, Boston, MA, 2002.

[14] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: Application in vlsi domain. *IEEE Transactions on VLSI Systems*, 20(1), 1999. A short version appears in the proceedings of DAC 1997.

[15] G. Karypis, E. Han, and V. Kumar. Chameleon: A hierarchical clustering algorithm using dynamic modeling. *IEEE Computer*, 32(8):68–75, 1999.

[16] G. Karypis and V. Kumar. hMETIS 1.5: A hypergraph partitioning package. Technical report, Department of Computer Science, University of Minnesota, 1998. Available on the WWW at URL *http://www.cs.umn.edu/~metis*.

[17] G. Karypis and V. Kumar. Multilevel k-way hypergraph partitioning. *VLSI Design*, 2000.

[18] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49(2):291–307, 1970.

[19] P.Fishburn. *Decision and Value Theory*. J.Wiley & Sons, New York, 1964.

[20] R.Keeney and H. Raiffa. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. J.Wiley & Sons, New York, 1976.

[21] L. Sanchis. Multiple-way network partitioning. *IEEE Trans. On Computers*, 38(1):62–81, 1989.

[22] K. Schloegel, G. Karypis, and V. Kumar. A new algorithm for multi-objective graph partitioning. In *Proceedings of EuroPar '99*, pages 322–331, 1999.

[23] S. Shekhar and D. R. Liu. Partitioning similarity graphs: A framework for declustering problmes. *Information Systems Journal*, 21(4), 1996.

[24] H. D. Simon and S.-H. Teng. How good is recursive bisection? Technical Report RNR-93-012, NAS Systems Division, NASA, Moffet Field, CA, 1993.

[25] M. Wang, S. K. Lim, J. Cong, and M. Sarrafzadeh. Multi-way partitioning using bi-partition heuristics. In *Proceedings of ASPDAC*, pages 441–446. IEEE, January 2000.

[26] S. Wichlund and E. J. Aas. On Multilevel Circuit Partitioning. In *Intl. Conference on Computer Aided Design*, 1998.

[27] P. Yu. *Multiple-Criteria Decision Making: Concepts, Techniques, and Extensions*. Plenum Press, New York, 1985.

[28] H. Zha, X. He, C. Ding, H. Simon, and M. Gu. Bipartite graph partitioning and data clustering. In *CIKM*, 2001.