Multilevel Hypergraph Partitioning: Application in VLSI Domain

George Karypis, Rajat Aggarwal, Vipin Kumar, and Shashi Shekhar
{karypis, rajat, kumar, shekhar}cs.umn.edu
University of Minnesota, Computer Science Department, Minneapolis, MN 55455

Abstract

In this paper, we present a new hypergraph partitioning algorithm that is based on the multilevel paradigm. In the multilevel paradigm, a sequence of successively coarser hypergraphs is constructed. A bisection of the smallest hypergraph is computed and it is used to obtain a bisection of the original hypergraph by successively projecting and refining the bisection to the next level finer hypergraph. We evaluate the performance both in terms of the size of the hyperedge cut on the bisection as well as run time on a number of VLSI circuits. Our experiments show that our multilevel hypergraph partitioning algorithm produces high quality partitioning in relatively small amount of time. The quality of the partitionings produced by our scheme are on the average 4% to 23% better than those produced by other state-of-the-art schemes. Furthermore, our partitioning algorithm is significantly faster, often requiring 4 to 15 times less time than that required by the other schemes. Our multilevel hypergraph partitioning algorithm scales very well for large hypergraphs. Hypergraphs with over 100,000 vertices can be bisected in a few minutes on today's workstations. Also, on the large hypergraphs, our scheme outperforms other schemes (in hyperedge cut) quite consistently with larger margins (9% to 30%).

1 Introduction

Hypergraph partitioning is an important problem and has extensive application to many areas, including VLSI design [2], efficient storage of large databases on disks, and data mining [14]. The problem is to partition the vertices of a hypergraph in k roughly equal parts, such that the number of hyperedges connecting vertices in different parts is minimized. Formally, a hypergraph $H = (V, E^h)$ is defined as a set of vertices V and a set of hyperedges E^h, where each hyperedge is a subset of the vertex set V.

Hypergraphs can be used to naturally represent a VLSI circuit [14]. Hypergraph partitioning has many applications including design packaging, HDL-based synthesis, design optimization, rapid prototyping, simulation, and testing. A high quality hypergraph partitioning algorithm greatly affects the feasibility, quality, and cost of the resulting system.

The problem of computing an optimal bisection of a hypergraph is NP-complete. However, because of the importance of the problem in many application areas, many heuristic algorithms have been developed. The survey by Alpert and Khang [2] provides a detailed description and comparison of various such schemes. In a widely used class of iterative refinement partitioning algorithms, an initial bisection is computed (often obtained randomly) and then the partition is refined by repeatedly moving vertices between the two parts to reduce the hyperedge-cut. These algorithms often use the Schweikert-Kernighan [21] (an extension of Kernighan-Lin (KL) [17] for hypergraphs), or the faster Fiduccia-Mattheyses (FM) [8] refinement heuristic to iteratively improve the quality of the partition. The partition produced by these methods is often poor, especially for larger hypergraphs, for a number of reasons. First, these methods choose vertices for movement based only upon local information. Second, if many vertices have the same gain, then the method offers no insight on which of these vertices to move [18]. Third, a hyperedge that has more than one vertices on both sides of the partition line does not influence the computation of the gain of vertices contained in it, making the gain computation quite inexact [5]. Hence, these algorithms have been extended in a number of ways [18, 20, 5, 6] that tend to enhance the performance of the basic KL/FM refinement algorithms, at the expense of increased run time.

Another class of hypergraph partitioning algorithms [7, 10, 9, 22] consists of two different phases. In the first phase, they cluster the hypergraph to form a small hypergraph and use the FM algorithm to bisect the small hypergraph. In the second phase, they use the bisection of this contracted hypergraph to obtain a bisection of the original hypergraph. The overall performance of such a scheme depends upon the quality of the clustering method. In many schemes, the projected partition is further improved using the FM refinement scheme [22].

Recently a new class of multilevel partitioning techniques was developed [3, 12, 11, 4]. These algorithms consist of three phases, namely, coarsening phase, initial partitioning phase, and uncoarsening and refinement phase. During the coarsening phase, a sequence of successively smaller (coarser) graphs is constructed; during the initial partitioning phase, a bisection of the coarsest graph is computed; and during the uncoarsening and refinement phase, the bisection is successively projected to the next level finer graph, and at each level an iterative refinement algorithm such as KL or FM is used to further improve the bisection. The various phases of multilevel bisection are illustrated in Figure 1. Karypis and Kumar extensively studied this paradigm in [16, 15] for partitioning of graphs. They presented new powerful graph coarsening schemes for which even a good bisection of the coarsest graph is a pretty good bisection of the original graph. This makes the overall multilevel paradigm

This work was supported by IBM Partnership Award, NSF CCR-9423082, Army Research Office contract DA/DAAH04-95-1-0538, and Army High Performance Computing Research Center, Army Research Laboratory cooperative agreement number DAAH04-95-2-0003/contract number DAAH04-95-C-0008. Related papers are available via WWW at URL: http://www.cs.umn.edu/~karypis

Design Automation Conference ®

0-89791-847-9/97/0006/$3.50 DAC 97 - 06/97 Anaheim, CA, USA

even more robust. Furthermore, it allows the use of simplified variants of KL and FM refinement schemes during the uncoarsening phase, which significantly speeds up the refinement without compromising the overall quality. METIS [16], a multilevel graph partitioning algorithm based upon this work, routinely finds substantially better bisections and is often two orders of magnitude faster than the hitherto state-of-the-art spectral-based bisection techniques for graphs.

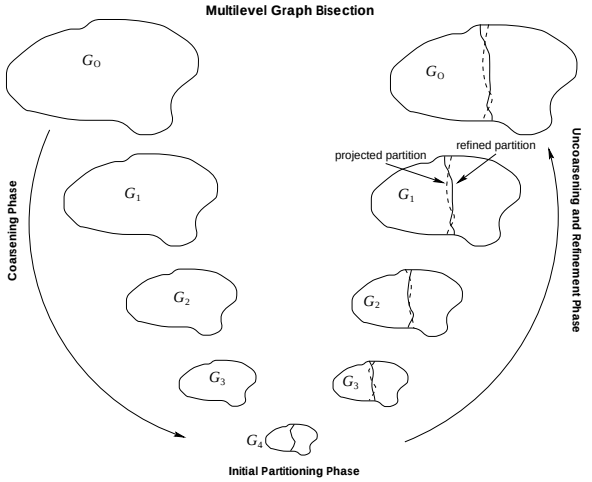

Figure 1: The various phases of the multilevel graph bisection.

The improved coarsening schemes of METIS work only for graphs, and are not directly applicable to hypergraphs. If the hypergraph is first converted into a graph (by replacing each hyperedge by a set of regular edges), then METIS [16] can be used to compute a partitioning of this graph. This technique was investigated by Alpert and Khang [1]. The conversion of a hypergraph into a graph by replacing each hyperedge by a clique does not result in an equivalent representation [14]. The fundamental problem associated with replacing a hyperedge by its clique, is that there exists no scheme to assign weight to the edges of the clique that can correctly capture the cost of cutting this hyperedge [13]. This hinders the partitioning refinement algorithm since vertices are moved between partitions depending on the reduction in the number of edges they cut in the converted graph, whereas the real objective is to minimize the number of hyperedges that are cut in the original hypergraph.

In this paper we present a multilevel hypergraph partitioning algorithm, hMETIS, that operates directly on the hypergraphs. A key contribution of our work is the development of new hypergraph coarsening schemes that allow the multilevel paradigm to provide high quality partitions quite consistently. The use of these powerful coarsening schemes also allow the refinement to be simplified considerably (even beyond the plain FM refinement), making the multilevel scheme quite fast.

We evaluate the performance both in terms of the size of the hyperedge cut on the bisection as well as run time on a number of VLSI circuits. Our experiments show that our multilevel hypergraph partitioning algorithm produces high quality partitioning in relatively small amount of time. The quality of the partitionings produced by our scheme are on the average 4% to 23% better that those produced by other state-of-the-art schemes [1, 5, 6, 19, 11]. The difference in quality over other schemes become even greater for larger hypergraphs. Furthermore, our partitioning algorithm is significantly faster, often requiring 4 to 10 times less time than that required by the other schemes. For many benchmark circuits in the well known ACM/SIGDA benchmark set [1], our scheme is able to find better partitionings than those reported in the literature for any other hypergraph partitioning algorithm.

[1] Available on the WWW at *http://vlsicad.cs.ucla.edu/~cheese/benchmarks.html*.

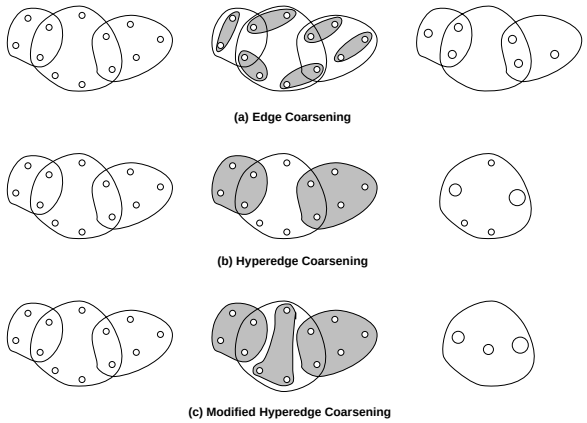

Figure 2: Various ways of matching the vertices in the hypergraph and the coarsening they induce.

2 Multilevel Hypergraph Bisection

We now present the framework of hMETIS, in which the coarsening and the refinement schemes treat hyperedges without any distortion. We have developed new algorithms for both of these phases, which in conjunction have the capability of delivering very good quality solutions. In the rest of this section, we briefly describe the algorithms used in the coarsening and the uncoarsening phases. Extensive descriptions can be found in [14].

Coarsening Phase The purpose of coarsening is to create a small hypergraph, such that a good bisection of the small hypergraph is not significantly worse than the bisection directly obtained for the original hypergraph. In addition to that, hypergraph coarsening also helps in successively reducing the size of the hyperedges. That is, after several levels of coarsening, large hyperedges are contracted to hyperedges connecting just a few vertices. This is particularly helpful, since refinement heuristics based on the Kernighan-Lin algorithm [17, 21, 8] are very effective in refining small hyperedges but are quite ineffective in refining hyperedges with a large number of vertices belonging to different partitions.

The group of vertices that are merged together to form single vertices in the next level coarse hypergraph can be selected in different ways. We have developed three different algorithms for coarsening. In *edge coarsening* (illustrated in Figure 2(a)) a heavy-edge maximal matching of the vertices of the hypergraph is computed to select the pairs of vertices. These vertices are then merged together. In *Hyperedge Coarsening*(illustrated in Figure 2(b)) an independent set of hyperedges is selected and the vertices that belong to these hyperedges are contracted together. This scheme gives preference to the hyperedges that have large weight and those that are of small size. In *Modified Hyperedge Coarsening*(illustrated in Figure 2(c)) after the hyperedges to be contracted have been selected using the hyperedge coarsening scheme, all the uncontracted hyperedges are considered again. And for each uncontracted hyperedge, the vertices that do not belong to any other contracted hyperedge are matched to be contracted together.

Uncoarsening and Refinement Phase We calculate the initial partitioning using "balanced" random bisection. This partitioning is then carried along in the uncoarsening phase. During the uncoarsening phase, a partitioning of the coarser hypergraph is used

to obtain a partitioning for the finer graph. This is done by successively projecting the partitioning to the next level finer hypergraph and using a partitioning refinement algorithm to reduce the cut and thus improve the quality of the partitioning. Since the next level finer hypergraph has more degrees of freedom, such refinement algorithms tend to improve the quality.

We have implemented two different partitioning refinement algorithms. The first is the *FM* algorithm [8] which repeatedly moves vertices between partitions in order to improve the cut. We have made two simplifications in FM, which are (i) we limit the maximum number of passes performed by the FM algorithm to only two and (ii) we stop each pass of the FM algorithm as soon as we have performed k vertex moves that did not improve the cut (this modification is called *early-exit FM* (FM-EE)). The second algorithm, called *Hyperedge Refinement(HER)*, moves groups of vertices between partitions so that an entire hyperedge is removed from the cut. Unlike FM, this algorithm has the capability to refine hyperedges that have many nodes on both sides of the partitioning boundary. Note that unlike FM, HER is not a hill-climbing algorithm, as it does not perform moves that can lead to a larger cut (*i.e.*, negative cut reduction).

3 Experimental Results

We experimentally evaluated the quality of the bisections produced by our multilevel hypergraph partitioning algorithm on a large number of hypergraphs that are part of the widely used ACM/SIGDA circuit partitioning benchmark suite. The characteristics of these hypergraphs are shown in Table 1. We performed all our experiments on an SGI Challenge that has MIPS R10000 processors running at 200Mhz, and all reported run-times are in seconds. All the reported partitioning results were obtained by forcing a 45–55 balance condition.

To compare the performance of the bisections produced by our multilevel hypergraph bisection and refinement algorithms, both in terms of bisection quality as well as runtime, we created Table 1. Table 1 shows the size of the hyperedge cut produced by our algorithms (hMETIS) and those reported by various previously developed hypergraph bisection algorithms. In particular, Table 1 contains results for the following algorithms: PROP [5], Opt. KLFM(scheme by Hauck and Borriello [11]) CLIP-PROP$_f$ [6], PARABOLI [19], and GMetis [1]. Note that for certain circuits, there are missing results for some of the algorithms. This is because no results were reported for these circuits. The column labeled "Best" shows the minimum cut obtained for each circuit by any of the earlier algorithms. Essentially, this column represents the quality that would have been obtained, if all the algorithms have been run and the best partition was selected.

The last two columns of Table 1 shows the partitionings produced by our multilevel hypergraph bisection and refinement algorithms. In particular, the column labeled "hMETIS-EE$_{20}$" corresponds to the best partitioning produced from 20 runs of our multilevel algorithm that uses early-exit FM during refinement (FM-EE). Of these twenty runs, ten runs are using hyperedge coarsening (HEC) and ten runs are using modified hyperedge coarsening (MHEC). The column labeled "hMETIS-FM$_{20}$" corresponds to the best partitioning produced from 20 runs when FM is used during refinement and coarsening is performed similarly to "hMETIS-EE$_{20}$". In both of these schemes, we used random initial partitionings during the initial partitioning phase.

To make the comparison with previous algorithms easier, we computed the total number of hyperedges cut by each algorithm, as well as the percentage improvement in the cut achieved by our algorithms over previous algorithms. This cut improvement was computed as the average improvement on a circuit-by-circuit level. Looking at these results, we see that both of our algorithms produce partitionings whose quality is better than that produced by any of the previous algorithms. In particular, hMETIS-EE$_{20}$ is 4.1% better than CLIP-PROP$_f$, 6.2% better than PROP, 9.9% better than Opt. KLFM, 10.0% better than GMetis, and 21.4% better as compared to PARABOLI. If all these algorithms are considered together, hMETIS-EE$_{20}$ is still better by 0.5%. Comparing hMETIS-EE$_{20}$ with hMETIS-FM$_{20}$ we see that hMETIS-FM$_{20}$ is about 1.1% better than hMETIS-EE$_{20}$, and about 1.7% better than all the previous schemes combined. In particular, hMETIS-FM$_{20}$ was able to improve the best-known bisections for 9 out of the 23 test circuits.

The last subtable of Table 1 shows the total amount of time required by the various partitioning algorithms. These run-times are in seconds on the respective architectures. Because of the difference in CPU speed at the various machines, it is hard to make direct comparisons. However, we tested our code on Sparc5 and we found that it requires about four times more time than when it is running on R10000. Taking into consideration a scaling factor of four, we see that both hMETIS-EE$_{20}$ and hMETIS-FM$_{20}$ require less time than either PROP, Opt. KLFM, CLIP-PROP$_f$, or PARABOLI. In particular, hMETIS-EE$_{20}$ is about four times faster than PROP, nine times faster than CLIP-PROP$_f$, ten times faster than PARABOLI, and fifteen times faster than Opt. KLFM. Comparing against GMetis, we see that hMETIS-EE$_{20}$ requires approximately the same time, whereas hMETIS-FM$_{20}$ is about twice as slow. Note that GMetis runs METIS 100 times on each graph but each of these runs is substantially faster than hMETIS, partly because METIS is a highly optimized code for graphs, and partly because coarsening and refinement for hypergraphs is more complex than the refinement schemes used in METIS. However, both hMETIS-EE$_{20}$ and hMETIS-FM$_{20}$ produce bisections that cut substantially fewer hyperedges than GMetis.

Furthermore, from Table 1, we can see that our scheme is more powerful relative to the other schemes for larger hypergraph. For example, restricting to only the larger hypergraphs (with 10K or more nodes) in the benchmark set, we find that hMETIS-FM$_{20}$ performs 4.8% better than the "Best".

4 Conclusions and Future Work

The multilevel hypergraph partitioning algorithm presented here is quite fast and robust. Even a single run of the algorithm is able to find reasonably good bisections. With a small number of runs (*e.g.*, 20) our algorithm is able to find better bisections than those found by all previously known algorithms for many of the well-known benchmarks. Also, our algorithm scales quite well for large hypergraphs. hMETIS is very fast, often requiring 4 to 15 times less time than that required by the other schemes. The quality of the bisection produced by hMETIS can be further improved by using the multi-phase refinement algorithm described in [14].

hMETIS is written in C language and is available at

http://www.cs.umn.edu/~karypis/metis

REFERENCES

[1] C. Alpert, L. Hagen, and A. Kahng. A hybrid multilevel/genetic approach for circuit partitioning. Technical Report, CS Dept., UCLA, Los Angeles, CA, 1996.

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

[3] T. Bui and C. Jones. A heuristic for reducing fill in sparse matrix factorization. In *6th SIAM Conf. Parallel Processing for Scientific Computing*, pages 445–452, 1993.

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

Benchmark	#Verices	#Hyper-edges	PROP	Opt. KLFM	CLIP-$PROP_f$	PARABOLI	GMetis	Best	hMETIS-EE_{20}	hMETIS-FM_{20}
balu	801	735	27	–	27	41	27	27	27	27
p1	833	902	47	–	51	53	47	47	52	50
bm1	882	903	50	–	47	–	48	47	51	51
t4	1515	1658	52	–	52	–	49	49	51	51
t3	1607	1618	59	–	57	–	62	57	58	58
t2	1663	1720	90	–	87	–	95	87	91	88
t6	1752	1541	76	–	60	–	94	60	62	60
struct	1952	1920	33	–	33	40	33	33	33	33
t5	2595	2750	79	–	77	–	104	77	71	71
19ks	2844	3282	105	–	104	–	106	104	107	106
p2	3014	3029	143	–	152	146	142	142	148	145
s9234	5866	5844	41	45	42	74	43	41	40	40
biomed	6514	5742	83	–	84	135	102	83	83	83
s13207	8772	8651	75	62	71	91	74	62	55	55
s15850	10470	10383	65	46	56	91	53	46	42	42
industry2	12637	13419	220	–	192	193	177	177	174	167
industry3	15406	21923	–	–	243	267	243	243	255	254
s35932	18148	17828	–	46	42	62	57	42	42	42
s38584	20995	20717	–	52	51	55	53	51	47	47
avq.small	21918	22124	–	–	144	224	144	144	136	130
s38417	23849	23843	–	–	65	49	69	49	52	51
avq.large	25178	25384	–	–	143	139	145	139	129	127
golem3	103048	144949	–	–	–	1629	2111	1629	1447	1445
Sum of Hyperedge-cuts										
5 circuits				251				248	226	226
16 circuits			1245					1143	1145	1127
16 circuits						3289		2949	2762	2738
22 circuits					1880			1797	1806	1778
23 circuits							4078	3426	3253	3223
hMETIS	**Quality improvement**									
EE_{20}			6.2%	9.9%	4.1%	21.4%	10.0%	0.5%		
FM_{20}			7.2%	9.9%	5.2%	22.4%	11.0%	1.7%	1.1%	
Runtime Comparison. The times are in seconds on the specified machines										
			Sparc5	Sparc IPX	Sparc5	Dec3000 500AXP	Sparc5		SGI R10000	SGI R10000
5 circuits				5606					95	125
16 circuits			2383						158	224
16 circuits						37570			874	1593
22 circuits					16206				445	637
23 circuits							3357		913	1654

Table 1: Performance of our multilevel hypergraph bisection algorithm (hMETIS) against various previously developed algorithms.

[5] S. Dutt and W. Deng. A probability-based approach to VLSI circuit partitioning. In *ACM/IEEE DAC*, 1996.

[6] S. Dutt and W. Deng. VLSI circuit partitioning by cluster-removal using iterative improvement techniques. In *Proc. Physical Design Workshop*, 1996.

[7] T. Bui et al. Improving the performance of the Kernighan-Lin and simulated annealing graph bisection algorithm. In *Proc. ACM/IEEE DAC*, pages 775–778, 1989.

[8] 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.

[9] Charles J. Alpert Lars W. Hagen and Andrew B. Kahng. A general framework for vertex orderings, with applications to netlist clustering. *to appear in IEEE Transactions on VLSI*.

[10] Lars Hagen and Andrew Kahng. A new approach to effective circuit clustering. In *Proceedings of IEEE International Conference on Computer Aided Design*, pages 422–427, 1992.

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

[12] Bruce Hendrickson and Robert Leland. A multilevel algorithm for partitioning graphs. Technical Report SAND93-1301, Sandia National Laboratories, 1993.

[13] E. Ihler, D. Wagner, and F. Wagner. Modeling hypergraphs by graphs with the same mincut properties. *Information Processing Letters*, 45(4), March 1993.

[14] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Hypergraph partitioning: Applications in VLSI domain. Technical Report TR-96-060, CS Dept., University of Minnesota, 1996.

[15] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. Technical Report TR 95-035, CS Dept., University of Minnesota, 1995. To appear in SIAM Journal of Scientific Computing.

[16] G. Karypis and V. Kumar. METIS: Unstructured graph partitioning and sparse matrix ordering system. Technical report, CS Dept., University of Minnesota, 1995.

[17] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Tech. Jour.*, 1970.

[18] B. Krishnamurthy. An improved min-cut algorithm for partitioning VLSI networks. *IEEE Trans. on Comp.*, C-33, May 1984.

[19] B. M. Riess, K. Doll, and F. M. Johannes. Partitioning very large circuits using analytical placement techniques. In *Proceedings ACM/IEEE DAC*, pages 646–651, 1994.

[20] Y. Saab. A fast and robust network bisection algorithm. *IEEE Transactions on Computers*, 44(7):903–913, 1995.

[21] D. G. Schweikert and B. W. Kernighan. A proper model for the partitioning of electrical circuits. In *Proc. ACM/IEEE Design Automation Conference*, pages 57–62, 1972.

[22] H. Shin and C. Kim. A simple yet effective technique for partitioning. *IEEE Transactions on VLSI Systems*, 1(3), 1993.