Hypergraph Partitioning With Fixed Vertices*

Andrew E. Caldwell, Andrew B. Kahng and Igor L. Markov

UCLA Computer Science Department, Los Angeles, CA 90095-1596

Abstract

We empirically assess the implications of fixed terminals for hypergraph partitioning heuristics. Our experimental testbed incorporates a leading-edge multilevel hypergraph partitioner [14] [3] and IBM-internal circuits that have recently been released as part of the ISPD-98 Benchmark Suite [2, 1]. We find that the presence of fixed terminals can make a partitioning instance considerably easier (possibly to the point of being "trivial"): much less effort is needed to stably reach solution qualities that are near best-achievable. Toward development of partitioning heuristics specific to the fixed-terminals regime, we study the pass statistics of *flat* FM-based partitioning heuristics. Our data suggest that with more fixed terminals, the improvements in a pass are more likely to occur near the beginning of the pass. Restricting the length of passes – which degrades solution quality in the classic (free-hypergraph) context – is relatively safe for the fixed-terminals regime and considerably reduces run time of our FM-based heuristic implementations. We believe that the distinct nature of partitioning in the fixed-terminals regime has deep implications (i) for the design and use of partitioners in top-down placement, (ii) for the context in which VLSI hypergraph partitioning research is pursued, and (iii) for the development of new benchmark instances for the research community.

1 Introduction

Hypergraph partitioning research in VLSI CAD has been primarily motivated by the gate-level top-down placement context, which in modern ASIC design methodology can demand extremely efficient and high-quality solutions for netlist sizes exceeding 1 million vertices. New heuristics for hypergraph partitioning are typically evaluated in the context of *free hypergraphs*, where all vertices are free to move into any partition [4, 2]. *Every benchmark, and every benchmark result reported in the literature, is for the free-hypergraph context.* Even when I/O pad locations are specified in the .vpnr or .yal source for early ACM/SIGDA benchmarks, the partitioning benchmarks (in .net/.are format; see [1]) do not indicate how these pads correspond to fixed vertices in partitions.

Our study is motivated by the following observation: *In top-down placement, the input to the partitioner is* **never** *a free hypergraph.* Rather, the input contains *fixed terminals* that arise from the chip I/Os or from the propagated terminals of other sub-problems in the partitioning hierarchy [6, 16]. The number of these fixed terminals can be estimated from Rent's rule [15, 5], which states that in a layout with Rent parameter p, on average a block of C cells will have $T = k \cdot C^p$ propagated or external terminals. This corresponds to a partitioning instance of $C+T$ vertices, of which T are fixed. Here, k is a constant equal to the average number of pins per cell, and is approximately 3.5 for modern designs; Rent parameter values for modern designs have been estimated at around 0.68 [5, 17]. Table 1 shows the maximum block sizes below which we expect all blocks (in a design with Rent parameter p) to have a given percentage of their vertices fixed.[1] We observe that even rather sizable sub-blocks of the design can be expected to have a high proportion of fixed terminals.

Rent Parameter	5%	10%	20%
$p = 0.60$	40992	7250	1281
$p = 0.65$	186943	25800	3561
$p = 0.70$	1413600	140250	13915

Table 1: Block sizes below which the expected number of fixed vertices due to propagated terminals will exceed a specified percentage (5%, 10% or 20%) of the total number of vertices in a top-down placement when the design has given Rent parameter p. We assume that the average pins per cell in the design is $k = 3.5$.

With this paper, we bring attention to the problem of partitioning with fixed terminals, and demonstrate that unique aspects of the fixed-terminals regime may require new partitioning heuristics Hence, the nature of partitioning in the fixed-terminals regime can have deep implications (i) for the design and use of partitioners in top-down placement, (ii) for the context in which VLSI hypergraph partitioning research is pursued, and (iii) for the development of new benchmark instances for the research community.

In Section 2 below, we empirically assess the implications of fixed terminals for hypergraph partitioning heuristics. Our experimental testbed incorporates a leading-edge multilevel [14] [3] hypergraph partitioner and IBM-internal circuits that have recently been released as part of the ISPD-98 Benchmark Suite [2, 1]. We conclude that the presence of fixed terminals can make a partitioning instance considerably easier (possibly to the point of being "trivial"): much less effort is needed to stably reach solution qualities that are near best-achievable. Section 3 presents early studies aimed at developing partitioning heuristics specific to the fixed-terminals regime. We study the pass statistics of *flat* FM-based partitioning heuristics, and demonstrate that with more fixed terminals, the improvements in a pass are more likely to occur near the beginning of the pass. A heuristic that restricts the length of passes – which would degrade solution quality in the classic (free-hypergraph) context – is relatively safe for the fixed-terminals regime and considerably reduces runtime of our FM-based implementations. Section 4 concludes with directions for future work.

2 Effect of Fixed Terminals on Instance Difficulty

We start by posing our motivating experimental questions, then describe our experimental testbed and protocol, followed by empirical results.

*Research supported by a grant from Cadence Design Systems, Inc.

[1] This assumes that the blocks are in "Region I" of the Rent parameter fit [15].

2.1 Experimental Questions

1. By how much can the presence of fixed terminals affect the solution quality achieved by modern partitioning heuristics?
2. Do partitioning instances with fixed terminals require less effort to "solve well" (with modern partitioning heuristics) than similar-complexity instances without fixed terminals?
3. Can guidelines be inferred as to the necessary effort required to achieve good partitioning solutions when a given proportion of the hypergraph vertices are fixed?

2.2 Experimental Testbed

Our experimental testbed contains implementations of common FM-based partitioning heuristics and standard partitioning benchmarks.

Partitioner

We use an internally developed partitioning engine that implements the *multilevel FM* approach described in [3] and [14]. Implementation details generally follow the parameters established in [3] (use of CLIP [7], heavy-edge matching, clustering ratio, etc.). The partitioning engine does not perform V-cycling as in [14], since V-cycling was determined to be a net loss in terms of overall cost-runtime profile of our partitioner.

The partitioning engine achieves solution quality and runtimes on a per-start basis that are somewhat better than those reported for ML_C [3] and *hMetis* [14] in the 1998 paper of Alpert [2] and on Alpert's web page [1]. This is confirmed by the experimental data reported in the next section.

Test Data

We have run experiments with the IBM01 through IBM05 test cases from the ISPD-98 Benchmark Suite developed by Alpert [2, 1]. We use actual areas of cells, and a 2% balance constraint. Because the cell areas vary considerably in the IBM benchmarks (there are often individual cells that occupy several percent of the total area [1]), there there is little point in doing unit-area studies for the real-life placement context. Moreover, tight balance constraints are more appropriate to the top-down cell placement application.

Experimental Protocol

In our experiments we choose vertices to fix at random from the set of all vertices in the netlist. We either (i) fix the chosen vertices independently into random partitions (**"rand"** in Figures 1 and 2), or (ii) fix the chosen vertices according to where they are assigned in the best min-cut solution we could find for the instance when no vertices were fixed (**"good"**). For each of the resulting four regimes we fix a number of vertices equal to 0%, 0.1%, 0.5%, 1.0%, 2.0%, 5%, 10%, 15%, 20%, 30%, 40% and 50% of the total number of vertices in the instance.[2] We apply the multilevel CLIP FM engine noted in the previous subsection. A single *trial* applies this partitioner to the given partitioning instance for 1, 2, 4 or 8 independent starts, and returns the best cutsize obtained as well as the number of CPU seconds used. (All CPU times are for a 140MHz Sun Ultra-1 workstation running Solaris2.6.) All of our data represent averages of 50 trials.

[2] In generating these instances, we *incrementally* fix additional vertices, e.g., all vertices fixed at 1.0% are also fixed at 2.0%.

2.3 Experimental Results

Figures 1 and 2 show detailed results for the IBM01 and IBM03 test cases, respectively. All data shown are for experiments where fixed vertices are chosen randomly from the set of all vertices in the instance.

- Each Figure contains six plots, with four traces in each plot corresponding to 1, 2, 4 and 8 starts of the multilevel partitioning engine.
- The upper ("good") row of each Figure gives data for the regime where all fixed vertices are consistent with the best solution that we know for the unconstrained (no fixed vertices) instance. The lower ("rand") row of each Figure gives data for the regime where the fixed vertices are randomly assigned to partitions.
- The left two plots in each Figure show the *raw solution costs* (best cutsize obtained with the given number of starts, averaged over 50 trials) versus the percentage of fixed vertices.
- Plots in the middle column in each Figure show the *normalized solution costs* versus the percentage of fixed vertices. In the "good" regime, the normalization is to a single constant value (since all instances have fixed vertices consistent with the same good solution), so the shape of the traces is similar to the plot of raw solution costs. However, in the "rand" regime, the raw solution costs increase drastically with the percentage of randomly chosen/fixed vertices, and each percentage of fixed vertices corresponds to a distinct partitioning instance. Thus, for each instance in the "rand" regime, we normalize solution costs to the best solution cost seen over all $(1+2+4+8)\times 50 = 750$ starts of the multilevel partitioner for that instance.
- The right two plots in each Figure show the *per-start CPU time* versus the percentage of fixed vertices.

We performed similar experiments where fixed vertices are chosen randomly from the set of identified I/Os (pads) in the netlist.[3] However, we do not discuss these results, for several reasons. First, the number of I/Os is typically very small (less than one percent of all vertices). Second, for those percentages of fixed vertices that could be chosen from I/Os we could find no difference in any experiment between fixing identified I/Os and fixing random vertices. Finally, for the vast majority of hierarchical block partitioning instances in top-down placement, the fixed terminals do not correspond to chip I/O pads anyway.

We also do not show results for the IBM02, IBM04 and IBM05 test cases. This is because the data looks essentially identical to what we have shown for IBM01 and IBM03. Indeed, we have been agreeably surprised by the consistency of our experimental results.

From the Figures, we make the following observations.

- The raw solution costs[4] indicate that as more fixed vertices are (randomly) selected and assigned to partitions, the achievable solution cost increases rapidly. This addresses the first experimental question: the presence of fixed vertices matters.
- The normalized solution costs indicate that if the netlist has many terminals fixed in partitions (which is is what we believe distinguishes real-life partitioning instances generated during top-down placement), then the partitioning problem is indeed "easy".

[3] When the fixed vertices are chosen from pads in the netlist, the percentage is limited by the total number of pads and we do not fix any further vertices.

[4] Note that these raw solution costs suggest that our multilevel partitioner is (at least) on par with [3] [14] in terms of solution quality.

Testcase	Percent Fixed Terminals						
	0%	5%	10%	15%	20%	30%	50%
IBM01	12.0,20.5%	12.6,10.8%	9.9,9.1%	8.9,7.6%	8.0,4.8%	6.1,2.6%	4.6,0.8%
IBM02	9.6,12.4%	8.3,10.0%	8.1,6.1%	7.7,3.7%	6.7,2.7%	6.7,2.4%	6.4,3.2%
IBM03	10.4,6.8%	9.8,6.5%	8.3,5.7%	7.6,4.8%	6.4,3.3%	6.4,4.1%	6.0,3.7%
IBM04	12.9,8.3%	10.9,7.0%	9.9,4.4%	7.4,3.5%	8.1,2.6%	7.4,3.5%	6.1,1.0%
IBM05	32.6,37.0%	16.4,5.3%	13.0,4.7%	10.1,4.0%	8.6,3.5%	7.9,2.3%	5.3,1.5%

Table 2: Average number of passes per run, and average percentage of nodes moved per pass (excluding the first pass), for 50 runs of LIFO-FM. Partitions are allowed to deviate from exact bisection by 2%.

  - When 0% of the vertices are fixed in partitions, more starts (e.g., 4 or 8) are required for the average best cutsize to approach the value that the multilevel partitioner is capable of achieving for the given instance.
  - When larger percentages of the vertices are fixed in partitions, fewer starts (e.g., 1 or 2) are required for the average best cutsize to approach the "good solution cost".
  - In the normalized traces, the curves are "flatter" (and there is less difference between the 1-start and 8-start traces) as the percentage of fixed vertices increases.
  - In all of our experiments, an instance with 20% or more vertices fixed is essentially solvable to very high quality in one or two starts, i.e., further starts are unnecessary. This suggests that most hierarchical block partitioning instances in placement are easy; recall Table 1 from Section 1.[5]

- Runtimes decrease substantially when the percentage of fixed vertices increases; this is expected since the partitioner has less freedom and a smaller number of movable vertices.

- Solution quality for "good" instances, and runtime for "rand" instances, are non-monotone in the percentage of fixed vertices. We suspect that this indicates "relatively overconstrained" instances where the inflexibility of the instance hurts the ability of the partitioner to find "trajectories to good solutions" more than it helps the partitioner by reducing the solution space. An interesting direction for future work is to attempt to demonstrate this effect. As discussed below, the data also suggest that current partitioning technology is not well-tuned to the fixed-terminals regime.

3 Toward Partitioners for the Fixed-Terminals Regime

We now describe preliminary explorations into partitioning methods designed specifically for the fixed-terminals regime. The first subsection presents motivating studies of pass-length statistics. The second subsection gives runtime and cutsize results for a new heuristic variant, using the same fixed-terminals instances described above.

3.1 FM Pass Structure With Fixed Terminals

A motivating observation is that in the absence of sufficient fixed terminals, FM may occasionally produce passes in which nearly every node is moved. Recall that during an FM pass, nodes are moved one at a time until each node has been moved; for bipartitioning, all nodes have been "flipped" when the end of the pass is reached. Then, the best solution found during the pass (i.e., best prefix of the move sequence) is restored. Any move "undone" in this process has essentially been wasted. Without terminals, FM will occasionally produce passes in which almost no moves are wasted – the pass flips almost all nodes between partition 0 and partition 1. However, if there are sufficiently many nodes adjacent to fixed terminals, such a "near-flip" is very unlikely to be improving. Table 2, documents the average number of passes per run, and the average percentage of nodes moved per pass: increasingly higher percentages of the moves in the FM passes are wasted as the proportion of fixed terminals increases. This strongly suggests that in the fixed-terminals regime, FM-style heuristics can profitably impose a hard cut-off on pass lengths.

3.2 FM Variants With Early-Stop Passes

Since the first FM pass traditionally begins with a random partitioning, many nodes will be moved, regardless of the number of fixed terminals. However, we may limit the number of moves per pass – after the first pass – in order to reduce overhead when the best solution found is near the beginning of the pass. Table 3 documents the effects on average cutsize and average CPU time for single LIFO FM starts, when FM passes are cut off after 50%, 25%, 10% or 5% of the moves have been made. For instances without sufficient terminals, early stopping has a detrimental effect on solution quality, but with sufficient terminals no effect on solution quality is seen. In all cases, limiting the number of moves in a pass improves runtime. solution quality. A surprising observation is that current partitioners appear to struggle when faced with only a small proportion (e.g., 5% or 10%) of fixed terminals. Because all terminals are fixed in a "good" location, and because fixed terminals are added only to produce problems with a higher percentage of fixed nodes, any solution for the cases of 20% or 0% fixed is also feasible for the case of 10% fixed. The fact that the partitioner produces better results for both the 20% and the 0% cases than for the 10% case may point to a failing of current partitioners.

4 Conclusions and Open Problems

We have empirically demonstrated a mismatch between the top-down placement context and current directions in VLSI CAD hypergraph partitioning research and benchmarking. We point out how easy the partitioning problem becomes when fixed terminals are present, and how strong this effect is. We believe that there is a great deal of work remaining to be done in the area of extremely fast partitioning for the fixed-terminals regime, i.e., the real-world placement context.

Our early efforts have entailed per-pass analyses of *flat* FM-based partitioning heuristics, confirming that the presence of fixed terminals limits the improvements in a pass to moves made at the beginning of the pass. Hard cut-offs on pass length – which degrades solution quality in the classical "free-hypergraph" context – is relatively safe in the presence of terminals, and considerably reduces runtime of our FM-based implementations.

An open and rather pragmatic issue is whether faster algorithms can be developed that exploit the presence of fixed terminals in applications such as top-down placement. Further analysis of the effects of fixed terminals may be useful. In particular, it is not yet clear how to measure the "strength" of fixed terminals, or alternatively the "degree of constraint" in particular problem instances. While our experiments fix random terminals from known hypergraphs where most vertices have low degree, it is always possible to fix vertices of very high degree to yield qualitatively different problem instances with similar numbers of fixed terminals. Indeed, a bipartitioning instance with arbitrary number/percent of fixed terminals can be represented by an equivalent instance with only two terminals, by clustering all terminals fixed in a given partition into

[5]The benefit from additional starts decreases more noticeably in the "rand" regime than in the "good" regime. Since propagated terminals are not likely assigned to their ideal locations, the benefit from starts in the top-down placement context is likely somewhere between the "rand" and "good" portraits.

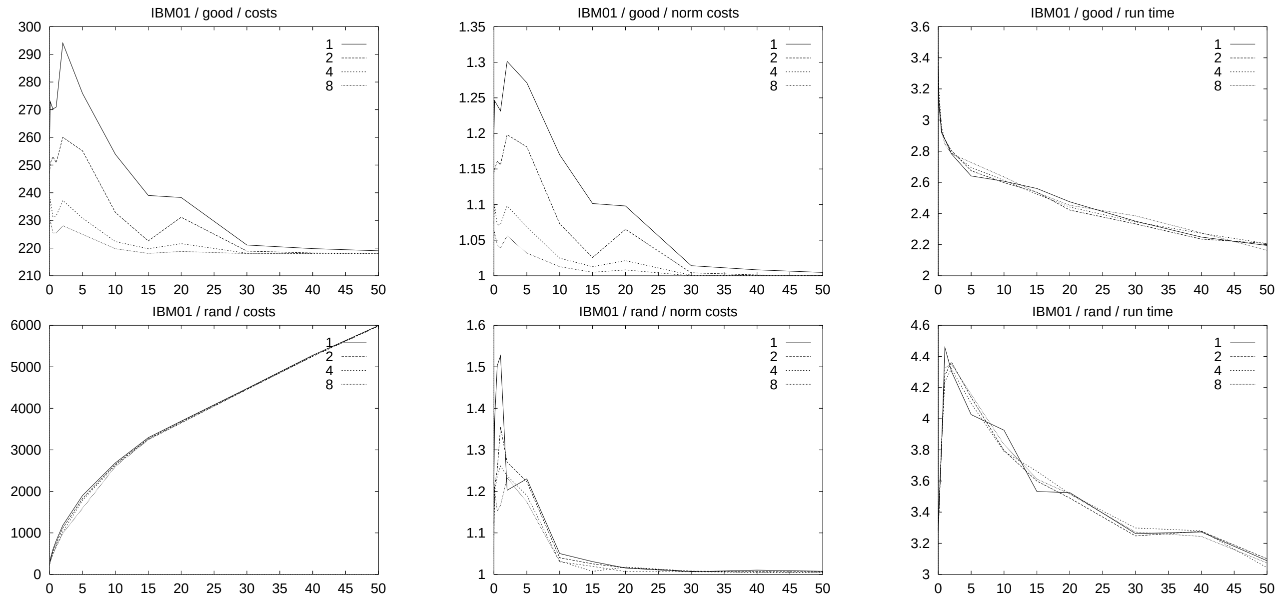

Figure 1: Experimental results for IBM01 test case, actual cell areas, 2% balance tolerance. The four traces in each plot correspond to 1, 2, 4 and 8 starts of the multilevel partitioner. We report raw best solution costs (left column), normalized best solution costs (middle column) and total CPU times (right column) for both the "good" (upper row) and "rand" (lower row) regimes. In all plots, the given parameter is plotted against the percentage of fixed vertices in the instance.

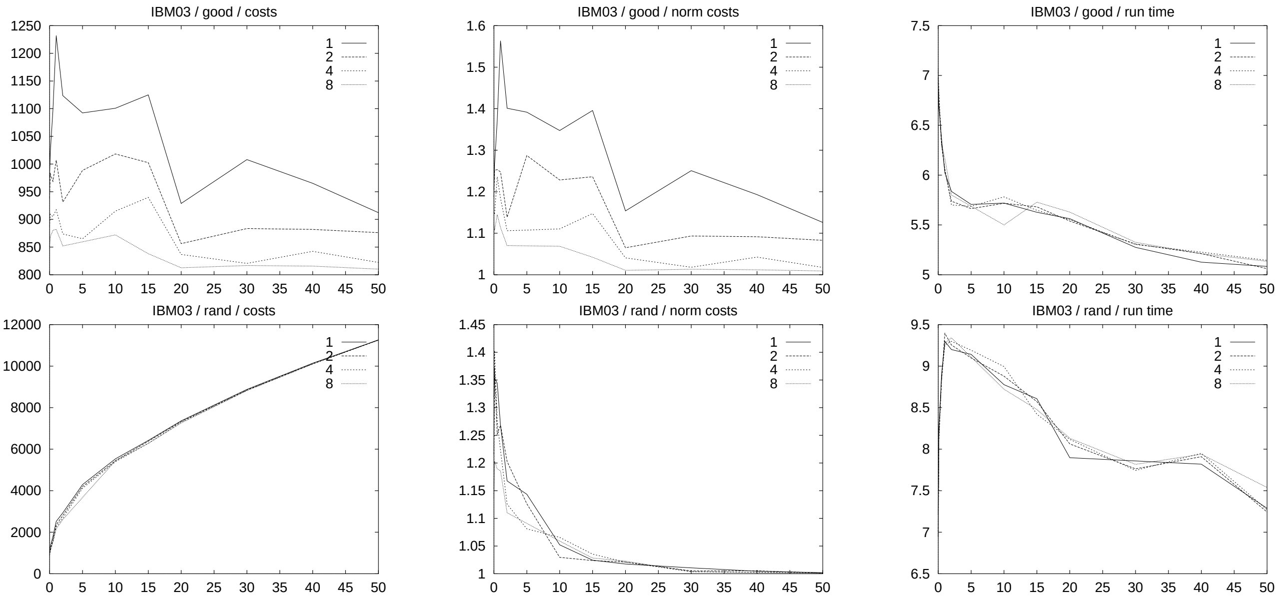

Figure 2: Experimental results for IBM03 test case, actual cell areas, 2% balance tolerance. The four traces in each plot correspond to 1, 2, 4 and 8 starts of the multilevel partitioner. We report raw best solution costs (left column), normalized best solution costs (middle column) and total CPU times (right column) for both the "good" (upper row) and "rand" (lower row) regimes. In all plots, the given parameter is plotted against the percentage of fixed vertices in the instance.

Testcase	Max Percent To Move	Percent Fixed Terminals						
		0%	5%	10%	15%	20%	30%	50%
IBM01	nolimit	596.2(2.81)	1274.7(2.8)	1041.8(2.03)	687.1(1.74)	513.8(1.51)	303.4(1.09)	247.0(0.724)
	50%	855.9(1.66)	1458.1(1.84)	1027.3(1.62)	720.3(1.35)	555.2(1.28)	290.3(0.999)	244.6(0.672)
	25%	959.4(1.49)	1702.8(1.5)	1249.1(1.26)	777.0(1.08)	521.7(0.98)	297.4(0.77)	245.0(0.593)
	10%	1233.6(1.47)	1811.4(1.39)	1435.3(1.18)	861.5(0.956)	514.7(0.832)	298.0(0.665)	242.8(0.496)
	5%	1533.9(1.61)	2154.7(1.41)	1707.7(1.21)	1235.9(0.992)	631.5(0.845)	305.4(0.639)	247.4(0.462)
IBM02	nolimit	515.1(3.5)	764.0(2.83)	1006.6(2.67)	1263.6(2.36)	1258.3(2.05)	1180.9(1.91)	1713.8(1.49)
	50%	547.0(2.45)	808.0(2.47)	1129.9(2.07)	1162.2(1.92)	1243.5(1.74)	1148.6(1.57)	1951.9(1.57)
	25%	621.4(2.03)	805.1(1.95)	1064.9(1.57)	1089.8(1.48)	1430.5(1.5)	1452.3(1.41)	1541.5(1.35)
	10%	718.5(1.95)	1222.9(1.72)	1280.3(1.47)	1299.6(1.28)	1280.6(1.23)	1206.8(1.2)	2134.3(1.14)
	5%	875.3(2.32)	1379.3(1.75)	1270.3(1.55)	1407.1(1.28)	1172.7(1.16)	1353.1(1.06)	1804.7(1.01)
IBM03	nolimit	1929.7(4.46)	2544.7(4.03)	2867.0(3.31)	3391.4(3.04)	1709.0(2.22)	1509.8(2.1)	1713.3(1.8)
	50%	2327.9(3.61)	2831.0(3.22)	2870.5(2.62)	3460.0(2.64)	1666.8(2.01)	1524.5(2.11)	1811.2(1.94)
	25%	2160.5(2.56)	2896.8(2.75)	2643.2(2.32)	3135.3(2.11)	1729.3(1.72)	1653.0(1.88)	1412.7(1.45)
	10%	2250.4(2.35)	2967.4(2.25)	3054.5(1.95)	3809.2(1.89)	1403.2(1.37)	1456.1(1.41)	1728.1(1.37)
	5%	2198.7(2.17)	2985.4(2.05)	3100.4(1.98)	3182.2(1.77)	1634.4(1.3)	1711.6(1.4)	1633.3(1.25)
IBM04	nolimit	2018.4(6.8)	2315.6(5.69)	1921.7(4.83)	2115.8(3.71)	1898.3(3.74)	1870.0(3.16)	991.6(2.2)
	50%	2411.2(5.02)	2181.8(4.57)	2173.0(3.92)	2005.3(3.17)	1920.3(3.11)	1623.2(2.88)	982.1(2.19)
	25%	2337.0(3.92)	2447.0(3.5)	2047.4(2.99)	1869.7(2.54)	1681.8(2.47)	1477.5(2.18)	973.3(1.75)
	10%	2646.1(3.46)	2829.6(3.75)	2175.7(2.71)	1813.9(2.22)	1999.3(2.12)	1764.6(1.99)	951.8(1.55)
	5%	2957.7(3.84)	3109.4(3.72)	2513.9(2.9)	2131.0(2.27)	2125.0(2.14)	2010.3(1.83)	989.4(1.41)
IBM05	nolimit	3455.6(18.7)	3324.3(9.53)	2461.4(7.21)	2087.3(5.46)	1972.1(4.48)	1854.3(4.07)	1793.5(2.16)
	50%	4726.8(7.96)	3254.0(6.43)	2486.2(4.94)	2104.3(4.25)	1979.4(3.83)	1857.6(3.28)	1793.3(2.23)
	25%	4954.4(6.31)	3336.3(5.99)	2432.0(4.37)	2093.5(3.38)	1970.3(3.11)	1856.2(2.57)	1793.2(1.75)
	10%	5234.6(6.31)	3528.6(4.72)	2545.3(3.93)	2121.0(2.93)	1991.8(2.45)	1850.6(2.25)	1793.0(1.48)
	5%	5730.7(6.51)	3676.0(5.99)	2576.1(4.06)	2137.7(3.09)	2010.5(2.45)	1849.8(2.06)	1794.0(1.47)

Table 3: Effects of cutting off all passes (after the first pass) at the given move limit during LIFO-FM partitioning. Partitions are allowed to deviate from exact bisection by 2%. Data is expressed as average cut(average CPU time). CPU seconds measured on a 300MHz Sun Ultra-10.

one single terminal. For common partitioning heuristics, such a representation is likely to be just as easy or hard as the original instance; we therefore need to quantify the "degree of constraint" in an invariant way. Additional points of interest in partitioning with fixed terminals include:

- developing interchange formats and benchmark suites that correspond to partitioning with fixed terminals at various levels of placement (we have made some efforts toward this end);
- determining whether multi-way partitioning is affected by fixed terminals similarly to bipartitioning; and
- confirming the existence of "relatively overconstrained" instances.

References

[1] C. J. Alpert, "Partitioning Benchmarks for VLSI CAD Community", `http://vlsicad.cs.ucla.edu/~cheese/benchmarks.html`

[2] C. J. Alpert, "The ISPD-98 Circuit Benchmark Suite", *Proc. ACM/IEEE International Symposium on Physical Design*, April 98, pp. 80-85. See errata at `http://vlsicad.cs.ucla.edu/~cheese/errata.html`

[3] C. J. Alpert, J.-H. Huang and A. B. Kahng,"Multilevel Circuit Partitioning", *ACM/IEEE Design Automation Conference*, pp. 530-533.

[4] C. J. Alpert and A. B. Kahng, "Recent Directions in Netlist Partitioning: A Survey", *Integration*, 19(1995) 1-81.

[5] J. A. Davis, V. K. De and J. D. Meindl, "A Stochastic Wire-Length Distribution for Gigascale Integration (GSI) - Part I: Derivation and Validation", *IEEE Transactions on Electron Devices*, vol. 45(3), pp. 580-589.

[6] A. E. Dunlop and B. W. Kernighan, "A Procedure for Placement of Standard Cell VLSI Circuits", *IEEE Transactions on Computer-Aided Design* 4(1) (1985), pp. 92-98

[7] S. Dutt and W. Deng, "VLSI Circuit Partitioning by Cluster-Removal Using Iterative Improvement Techniques", *Proc. IEEE International Conference on Computer-Aided Design*, 1996, pp. 194-200

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

[9] M. R. Garey and D. S. Johnson, "Computers and Intractability, a Guide to the Theory of NP-completeness", W. H. Freeman and Company: New York, 1979, pp. 223

[10] M. K. Goldberg and M. Burstein, "Heuristic Improvement Technique for Bisection of VLSI Networks", *IEEE Transactions on Computer-Aided Design*, 1983, pp. 122-125.

[11] S. Hauck and G. Borriello, "An Evaluation of Bipartitioning Techniques", *IEEE Transactions on Computer-Aided Design* 16(8) (1997), pp. 849-866.

[12] D. J. Huang and A. B. Kahng, "Partitioning-Based Standard Cell Global Placement with an Exact Objective", *Proc. ACM/IEEE International Symposium on Physical Design*, 1997, pp. 18-25.

[13] B. W. Kernighan and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs", *Bell System Tech. Journal* 49 (1970), pp. 291-307.

[14] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel Hypergraph Partitioning: Applications in VLSI Design", *Proc. ACM/IEEE Design Automation Conference*, 1997, pp. 526-529.

[15] B. Landman and R. Russo, "On a Pin Versus Block Relationship for Partitioning of Logic Graphs", *IEEE Transactions on Computers* C-20(12) (1971), pp. 1469-1479.

[16] P. R. Suaris and G. Kedem, "Quadrisection: A New Approach to Standard Cell Layout", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 1987, pp. 474-477.

[17] D. Sylvester and K. Keutzer, "Getting to the Bottom of Deep-Submicron", to appear in *Proc. IEEE Intl. Conference on Computer-Aided Design*, November 1998.