# Partitioning With Terminals: A "New" Problem and New Benchmarks*

C. J. Alpert†, A. E. Caldwell, A. B. Kahng and I. L. Markov

UCLA Computer Science Dept., Los Angeles, CA 90095-1596 USA
†IBM Austin Research Laboratory, Austin, TX 78660 USA

**Abstract**

The presence of *fixed terminals* in hypergraph partitioning instances arising in top-down standard-cell placement makes such instances qualitatively different from the *free hypergraphs* that have driven the past two decades of VLSI CAD partitioning research. In this paper we empirically show that with fixed terminals in the instance, less effort is needed to stably reach a given solution quality. We then develop new benchmark formats that flexibly capture the presence of terminals and any geometric embedding information associated with the partitioning instance. Our new formats not only allow modeling of top-down placement, but also enable study of placement-specific partitioning objectives, e.g., based on net bounding boxes and Steiner tree estimators. Finally, we develop a new suite of partitioning benchmarks with fixed terminals, based on the actual placement data from the IBM-internal circuits released in the ISPD-98 Benchmark Suite [1, 2]. A set of partitioning results is presented with runtime regimes appropriate to the placement use model, as a baseline for future efforts in the research community.

## 1 Introduction

The *min-cut hypergraph partitioning problem* seeks an assignment of the vertices of a vertex- and hyperedge-weighted hypergraph to a prescribed number of *partitions*, such that the number of hyperedges with vertices in more than one partition is minimized. Hypergraph partitioning heuristics for VLSI CAD have for nearly two decades constituted an extremely active and fast-moving field of research. For example, less than 18 months ago, the state of the art was defined by $ML_C$ [3] and *hMetis*1.0 [13]. Today, the leading partitioner, *hMetis*1.5.3 [13], obtains over 10% cutsize reduction and is approximately 25% faster when compared to *hMetis*1.0 on the ISPD-98 benchmark suite [2].

Published hypergraph partitioning heuristics have always been evaluated according to their performance on *free hypergraphs*, i.e., instances where all vertices are free to move into any partition [4, 2]. In the VLSI CAD literature, every partitioning benchmark, and every benchmark result, is for the free-hypergraph context. Although I/O pads may be specified in original benchmark data with their locations, partitioning benchmark formats *.net/.netD* and *.are* [1] are not able to express any mapping between pads and fixed vertices in partitions.

Our work addresses a critical disconnect between (i) the benchmarks used in VLSI CAD hypergraph partitioning research and (ii) the actual instances found in the application domain for partitioning heuristics. Without question, the key driver for research in *large-scale* hypergraph partitioning has been top-down standard-cell placement. However, in the top-down placement domain, the input to the partitioner is never a free hypergraph: rather, the input always contains *fixed terminals* that arise from the chip I/Os or from the propagated terminals [7, 16] of other sub-problems in the partitioning hierarchy.

Formally, a *fixed terminal* is a vertex whose partition assignment is prescribed as part of the input specification. (Other vertices in the input are *movable*.) The number of fixed terminals in instances arising from top-down placement is quite large. This can be seen from Rent's rule [15, 6], 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, where $k$ is a constant equal to the average number of pins per cell. Such a block will induce a partitioning instance with $C + T$ vertices, of which $T$ are fixed.[1] To our knowledge, neither the effect of fixed terminals on the difficulty of partitioning instances, nor the appropriate choice of heuristics in the presence of fixed terminals, has yet been addressed in the partitioning literature.

In this paper, we make the following contributions.

- In Section 2, we empirically show that with more fixed terminals, less effort is needed to stably reach best-seen solution qualities.
- In Section 3 and the Appendix, we develop new benchmark formats that flexibly capture the presence of terminals and geometric embedding information associated with the partitioning instance. Our new formats extend the established `.netD` and `.are` formats and allow a variety of new uses, such as partitioning with wirelength estimators based on net bounding boxes or Steiner trees.
- In Section 4, we develop a new suite of partitioning benchmarks with fixed terminals, based on the actual placements of the IBM-internal circuits released in the ISPD-98 Benchmark Suite [2, 1]. We additionally present partitioning results with

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

[1] Modern designs have estimated Rent parameter values of around 0.68 [6, 17].

runtime regimes appropriate to the placement use model, as a baseline for future efforts in the research community.

## 2 Effect of Fixed Terminals on Instance Difficulty

In this section, we study the effect of fixed terminals on the performance of modern partitioning heuristics. We find that partitioning instances with fixed terminals require less effort to "solve well" than similar-complexity instances without fixed terminals.

### Partitioning Testbed

Our experiments use an internally developed partitioner that implements the *multilevel FM* approach described in [3] and [13]. Implementation details generally follow [3] (use of CLIP [8], heavy-edge matching, clustering ratio, etc.), except that (i) the partitioner does not perform V-cycling as in [13], since V-cycling was determined to be a net loss in terms of overall cost-runtime profile, and (ii) the clustering (heavy-edge matching) implementation can freely mix fixed vertices ("terminals") and movable vertices as long as all fixed vertices in a cluster have compatible partition assignments except that terminals can not be clustered together. As can be seen in Figures 1 and 3 below, the partitioner achieves solution quality and runtimes on a per-start basis that are somewhat better than those reported for $ML_C$ [3] and *hMetis*1.0 [13] in the 1998 paper of Alpert [2] and on Alpert's web page [1].

### Test Data and Experimental Protocol

To reflect the top-down placement context, our experiments use the test circuits from ISPD-98 Benchmark Suite [1, 2] with actual areas of cells and a 2% balance constraint.

- Randomly chosen vertices are fixed according to either of two regimes: (i) independently in random partitions (**"rand"**), or (ii) according to where they are assigned in the best min-cut solution we could find for the instance when no vertices were fixed (**"good"**). [2]
- For each of the two 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. The vertices are fixed incrementally, e.g., all vertices fixed at 1.0% are also fixed at 2.0%.
- A single *trial* applies the 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 (140MHz Sun Ultra-1). All data represents averages of 50 trials.

### Experimental Results

Figures 1-3 give detailed results for the IBM01 test case. Results for other ISPD-98 circuits were qualitatively identical.

Figure 1 shows the *raw solution costs* (best cutsize obtained with the given number of starts, averaged over 50 trials) plotted against the percentage of fixed vertices. On the left ("good") side are data for the regime where all fixed vertices are consistent with the best solution that we know for the unconstrained (no fixed vertices) instance. On the right ("rand") side are data for the regime where the fixed vertices are randomly chosen and randomly assigned to partitions. The four traces in each plot correspond to 1, 2, 4 and 8 starts of the multilevel partitioner. We see from these raw solution costs that our partitioner is on par with those of [3] [13]. As more vertices are (randomly) selected and fixed in partitions, achievable solution cost increases rapidly, as would be expected since the partitioning is more constrained.

Figure 2 plots 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 normalized solution costs indicate that if the netlist has many terminals fixed in partitions – as in real-life instances generated during top-down placement – the partitioning problem becomes "easy". For example, 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. On the other hand, 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". The normalized traces also "flatten" (and there is less difference between the 1-start and 8-start traces) as the percentage of fixed vertices increases. We observe that 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 additional starts in the top-down placement context is likely somewhere between the "rand" and "good" portraits.

Finally, Figure 3 plots the *per-start CPU time* (140MHz Sun Ultra-1 seconds) versus the percentage of fixed vertices. 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. Thus, fixed vertices make the problem "easy" not only in the sense of number of starts required for stably good solution quality, but also in the sense of the CPU time per start.

Overall, our experiments indicate that, on average, only two starts of a multilevel FM partitioner achieve very high quality solutions of partitioning instances with 20% or more of fixed vertices. Further starts are not as productive and often unnecessary. In light of the parameters of new benchmark instances and the baseline partitioning results given below, this suggests that most hierarchical block partitioning instances in placement are easy. Hence, we believe that a key direction for future algorithm research will focus on better runtime-quality tradeoffs in the regime of very small runtimes.

## 3 New Partitioning Benchmark Formats

This section describes new benchmark formats that are presented in detail in the Appendix. In addition to fixed terminals, they capture the geometric embedding information that reflects intermediate states in top-down placement. Each benchmark is comprised of

[2] We have also performed identical experiments where the fixed vertices are chosen randomly from the set of identified I/Os (pads) in the netlist. In such a context, the proportion of fixed vertices is bounded by the total number of pads — typically less than one percent of the netlist. 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. Thus, as far as we can tell, fixing random vertices does not lead to "unrealistic" instances or wrong conclusions: indeed, for the vast majority of hierarchical block partitioning instances in top-down placement, the fixed terminals do not correspond to chip I/Os.

Results for IBM01 test case, actual cell areas, 2% balance tolerance. The four traces correspond to 1, 2, 4 and 8 starts of the multilevel partitioner. The x-axis represents the percentage of fixed vertices in the instance.

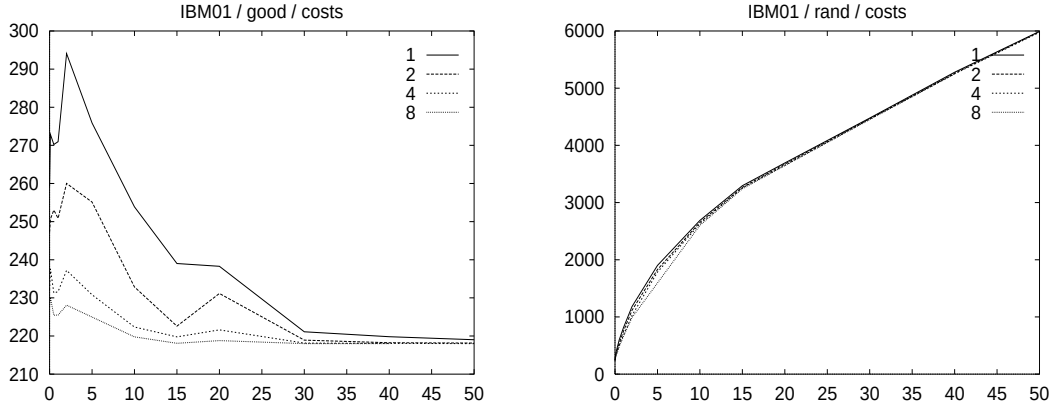


Figure 1: Raw best solution costs for both the “good” (left) and “rand” (right) regimes.

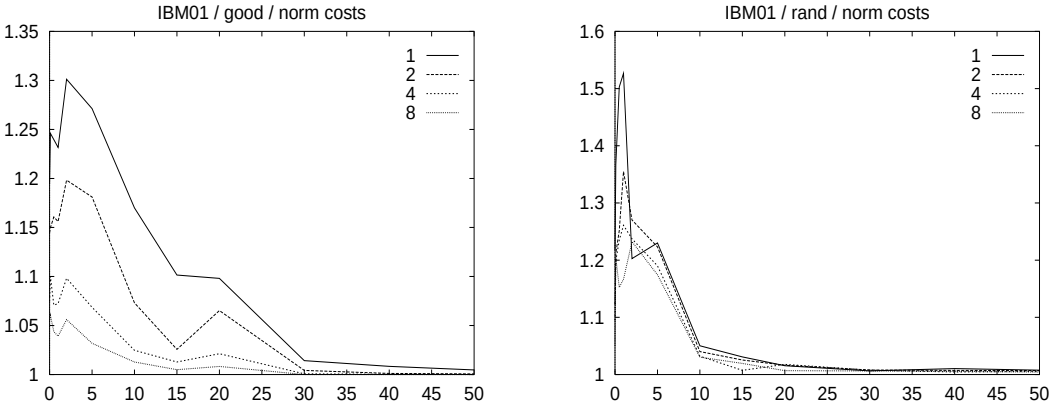


Figure 2: Normalized best solution costs for both the “good” (left) and “rand” (right) regimes.

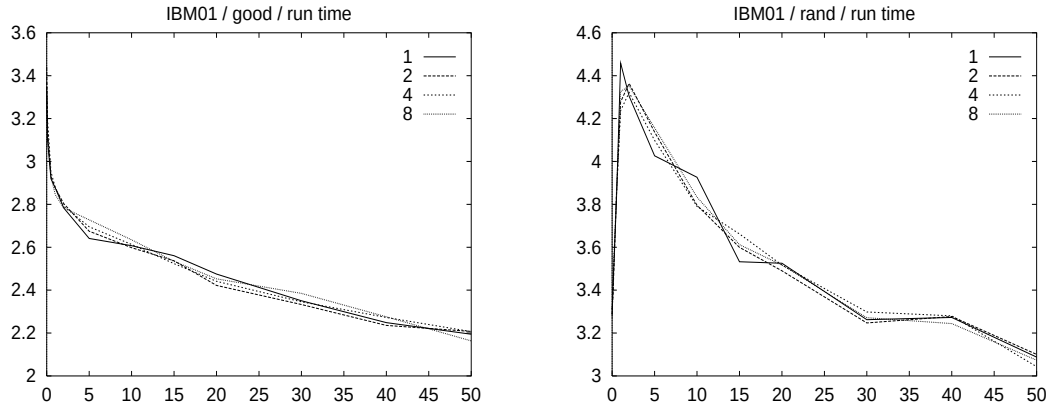


Figure 3: Per-start CPU times for both the “good” (left) and “rand” (right) regimes.

four files: a `.netD` file, a `.are` file, a UCLA `.fix` file, and a UCLA `.blk` file.

The `.netD` format was proposed at ISPD-98 [2] as an improvement to the `.net` format. The `.net` format and `.are` formats are "folklore" (dating from research by Wei and Cheng at UCSD circa 1990) and are retrospectively formalized, along with the `.netD` format, in [1]. Terminals are described in `.netD` format as "pads".

The new UCLA `.fix` format specifies fixed assignments of vertices to partitions, which can also describe terminal propagation. A `.fix` file can be used without an accompanying `.blk` file if the partitioning objective does not account for the geometry of partitions.[3]

The new UCLA `.blk` format specifies geometric attributes and capacities of multiple partitions, i.e., rectangular *blocks*. A `.blk` file does not depend on a `.fix` file: in combination with a `.netD` file, it is sufficient to specify an instance as long as no terminals are fixed.

The new formats feature:

- Multiple partition geometries and capacities, fixed modules and terminal propagation, in virtually any combination. Sufficiently intelligent parsers will not require redundant information.
- Flexible balance constraints represented using either absolute or relative (percentage) semantics.
- Straightforward facilities to represent "multi-balanced" partitioning problems where each module supplies the same number ($k > 1$) of resource types.[4] A corresponding set of $k$ capacities and tolerances must be specified for each partition. In a new "multi-area" `.areM` file type, each "area" corresponds to a given resource type; this is a straightforward extension of the `.are` file format with multiple module "areas" repeated on the same line.
- Flexible assignment of fixed terminals to partitions, which enables study of placement-specific partitioning objectives.[5] Terminals can be assigned to regions or to exact locations (via degenerate regions). Terminals can also be fixed in more than one partition while still retaining their "atomic" nature, i.e., the multiple assignment is interpreted as an "or". For example, a propagated terminal can be fixed in the two left-side quadrants of a quadrisection instance, so that the partitioner is free to assign it to either left-side quadrant.

## 4 New Benchmarks and Results for Calibration

To develop a new benchmark suite for partitioning with fixed terminals, the IBM Corporation has supplied $(x, y)$ location data for each cell and pad, corresponding to the actual placements of circuits in the ISPD-98 Benchmark Suite. From these placements, we develop a partitioning instance with fixed terminals as follows.[6]

A *block* is defined by a rectangular axis-parallel bounding box. An axis-parallel *cutline* bisects a given block. Each cell contained in the block induces a *movable vertex* of the hypergraph. Each pad adjacent to some cell in the block induces a zero-area *terminal* vertex of the hypergraph, fixed in the closest partition; adjacent cells not in the block similarly induce terminal vertices. From the placement of each IBMxx benchmark circuit, we extract four benchmark netlists IBMxxA - IBMxxD, each with two sets of terminal assignments (corresponding to vertical and horizontal cutlines) for partitioning with terminals. Each partitioning instance is named with the level at which it occurs (L0,L1,etc.) and the partitioning choices at higher levels which define it.[7]

- In IBMxxA_L0, the bounding box contains the entire core region, including the entire netlist with all cells and I/O pads. There are few fixed terminals, and they tend to be uniformly distributed.
- In IBMxxB_L1_V0 (IBMxxC_L1_V1), the bounding box contains the left (right) half of the core region. Fixed terminals in these benchmarks are unevenly distributed because part of the block bounding box is on the core boundary, and part is internal to the core.
- In IBMxxD_C11-33, the bounding box contains the "middle quarter" of the entire core region.[8]

| Circuit | Cells | Pads | ExtNets | Nets | Pins | Max% |
|---|---|---|---|---|---|---|
| IBM01A_L0 | 12506 | 246 | 246 | 14111 | 50566 | 6.37 |
| IBM01B_L1_V0 | 6388 | 1392 | 761 | 7384 | 27236 | 9.34 |
| IBM01C_L1_V1 | 6121 | 1377 | 763 | 7370 | 26951 | 10.03 |
| IBM01D_L3_C11-33 | 6739 | 2155 | 1227 | 7330 | 28661 | 9.54 |
| IBM06A_L0 | 32332 | 166 | 166 | 34826 | 128182 | 13.56 |
| IBM06B_L1_V0 | 13245 | 4360 | 1867 | 14786 | 58809 | 17.28 |
| IBM06C_L1_V1 | 19094 | 4086 | 1851 | 21824 | 81997 | 16.11 |
| IBM06D_L3_C11-33 | 10314 | 7553 | 3482 | 12438 | 55640 | 18.88 |
| IBM09A_L0 | 53110 | 285 | 285 | 60902 | 222088 | 5.42 |
| IBM09B_L1_V0 | 23461 | 28764 | 40003 | 47740 | 192358 | 5.49 |
| IBM09C_L1_V1 | 29649 | 23211 | 40038 | 53040 | 204928 | 5.45 |
| IBM09D_L3_C11-33 | 25099 | 27303 | 40137 | 49200 | 195924 | 5.50 |
| IBM10A_L0 | 68685 | 744 | 744 | 75196 | 297567 | 4.80 |
| IBM10B_L1_V0 | 25467 | 4971 | 3459 | 30072 | 115065 | 6.91 |
| IBM10C_L1_V1 | 43231 | 5260 | 3506 | 48246 | 197408 | 5.78 |
| IBM10D_L3_C11-33 | 26954 | 9376 | 6556 | 31529 | 129409 | 6.09 |
| IBM11A_L0 | 70152 | 406 | 406 | 81454 | 280786 | 4.48 |
| IBM11B_L1_V0 | 33506 | 4970 | 3323 | 40623 | 142760 | 6.40 |
| IBM11C_L1_V1 | 36646 | 5036 | 3291 | 43935 | 152703 | 6.29 |
| IBM11D_L3_C11-33 | 30971 | 9968 | 6068 | 36273 | 129724 | 6.92 |
| IBM12A_L0 | 70439 | 637 | 637 | 77240 | 317760 | 6.43 |
| IBM12B_L1_V0 | 38904 | 5233 | 3669 | 43660 | 179596 | 9.17 |
| IBM12C_L1_V1 | 31544 | 5218 | 3610 | 36907 | 152863 | 9.90 |
| IBM12D_L3_C11-33 | 27490 | 12411 | 8101 | 33215 | 142965 | 8.59 |
| IBM13A_L0 | 83709 | 490 | 490 | 99666 | 357075 | 4.23 |
| IBM13B_L1_V0 | 40582 | 6589 | 3578 | 49757 | 179870 | 6.09 |
| IBM13C_L1_V1 | 43127 | 5895 | 3586 | 53246 | 196549 | 6.41 |
| IBM13D_L3_C11-33 | 37193 | 12745 | 7754 | 45756 | 171864 | 6.42 |
| IBM16A_L0 | 182980 | 504 | 504 | 190048 | 778823 | 1.89 |
| IBM16B_L1_V0 | 99102 | 9083 | 5200 | 103816 | 432680 | 2.88 |
| IBM16C_L1_V1 | 83884 | 10491 | 5211 | 91190 | 375500 | 2.81 |
| IBM16D_L3_C11-33 | 65041 | 18326 | 8581 | 70899 | 300966 | 2.82 |
| IBM17A_L0 | 184752 | 743 | 743 | 189581 | 860036 | 0.94 |
| IBM17B_L1_V0 | 100763 | 10834 | 6684 | 106877 | 485785 | 1.58 |
| IBM17C_L1_V1 | 84015 | 12290 | 6716 | 89052 | 409926 | 1.19 |
| IBM17D_L3_C11-33 | 51714 | 23033 | 11444 | 58988 | 263516 | 2.10 |

Table 1: Parameters of new benchmarks with fixed terminals.

Parameters of the resulting instances are summarized in Table 1. Complete .netD, .are, .fix, and .blk files for these test cases are

[3] Partition capacities will need to be set by the user.

[4] A hypothetical example with $k = 3$ might include cell area, cell pin count, and cell power dissipation resource types — all of which must be evenly distributed between the partitions.

[5] E.g., based on net bounding boxes and Steiner tree estimators etc.

[6] Because the circuits from the original ISPD-98 Benchmark Suite have been placed by different flows throughout IBM, the intermediate states of the placement process are not available as sources from which partitioning benchmarks with fixed terminals may be derived.

[7] For instance, L1_V0 is the left block of a top-level vertical bisection.

[8] E.g., if the core region is defined by $(xmin, xmax, ymin, ymax)$ then we choose the bounding box defined by $(\frac{3 \cdot xmin + xmax}{4}, \frac{xmin + 3 \cdot xmax}{4}, \frac{3 \cdot ymin + ymax}{4}, \frac{ymin + 3 \cdot ymax}{4})$. Fixed terminals in these benchmarks are dense but uniformly distributed. The use of the "middle quarter" is consistent with "cycling and overlapping" [11] and routing hotspot removal techniques that are part of standard partitioning-based placement approaches.

available at http://vlsicad.cs.ucla.edu/benchmarks/ispd99/ . Note that Table 1 shows the size of the largest cell in the instance as a percentage (Max%) of the total cell area. Also, both (i) the number of "pads", and (ii) the number of external nets (nets that are incident to at least one "pad") are given. Our construction creates more "pad" vertices in the hypergraph than there are external nets (the latter correspond to propagated terminals in the sense of [7] or Rent's rule analysis). However, since "pads" according to either definition would be introduced with zero area, the partitioning problem is not affected. To verify that the numbers of external nets in our benchmarks are reasonable, Table 2 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. This calculation is based on the equation $T = k \cdot C^p$ (which applies to blocks that are in "Region I" of the Rent parameter fit [15]), with $k = 3.5$ (average pins per cell) and $p = 0.55, 0.60, 0.65, 0.70$. We see that the numbers of external nets in our benchmarks are consistent with Rent parameter values for industrial circuits (between $p = 0.55$ and $p = 0.75$) that have been reported in the literature.

| % Terminals | $p = 0.55$ | $p = 0.60$ | $p = 0.65$ | $p = 0.70$ |
|---|---|---|---|---|
| 5% | 12595 | 40996 | 186942 | 1413600 |
| 10% | 2699 | 7247 | 25800 | 140246 |
| 15% | 1096 | 2629 | 8100 | 36301 |
| 20% | 578 | 1281 | 3560 | 13914 |
| 25% | 352 | 733 | 1882 | 6613 |
| 30% | 234 | 464 | 1117 | 3601 |
| 35% | 166 | 316 | 719 | 2154 |

Table 2: Block sizes (# of cells) below which the expected number of fixed vertices due to propagated terminals will exceed a specified percentage (5%, 10%, 15%, ..., 30%) of the total number of vertices in a partitioning instance arising in top-down placement when the design has given Rent parameter $p$. We assume the average number of pins per cell is $k = 3.5$, and that the block is in Region I of the Rent's rule fit.

Finally, we have run what we believe to be the current leading hypergraph partitioning tool, hMetis-1.5.3 (publicly released November 1998 [13]), on our new benchmarks to establish a baseline for future research. Table 3 shows the minimum and average cutsize results obtained over 50 independent runs of hMetis-1.5.3, using 1, 2 or 4 starts per run. We see that the benefit of additional starts decreases as the proportion of fixed terminals increases, confirming the studies of Section 2 above. The Tables also report CPU times measured in seconds on a 140 MHz Sun Ultra-1.

## 5 Conclusions

The VLSI CAD partitioning literature has never evaluated its heuristics on instances with fixed terminals, even though the driving application (top-down standard-cell placement) always induces instances with fixed terminals. In particular, no standard benchmark format today captures the presence of fixed terminals.

We have experimentally shown that, on average, only two starts of a multilevel FM partitioner achieve very high quality solutions of partitioning instances with sufficiently many fixed vertices, while further starts are often unnecessary. A detailed explanation of this phenomenon and the design of algorithms to exploit it are open research questions. To facilitate such research, we propose new benchmark formats and instances that capture geometric embedding information associated with the partitioning instance, e.g. terminal locations in top-down placement. The new suite of partitioning benchmarks with fixed terminals is derived from the actual placements of the IBM circuits known to the partitioning community as the ISPD-98 Benchmark Suite [2, 1]. We present baseline partitioning results for runtime regimes appropriate to the placement use model.

Goals of our ongoing research include (i) more formal analyses of the effect of fixed terminals on (move-based) partitioning heuristics and reachability within the solution space, (ii) new heuristics aimed at the regime of very short runtimes and a relatively large proportion of fixed terminals, and (iii) clustering heuristics (e.g., for multilevel FM variants) appropriate to the fixed-terminals context.

## Appendix: New Benchmark File Formats

### UCLA `.blk` File Format

Empty lines and lines starting with pound sign '#' are ignored by the parser.[9]. The first non-ignored line is reserved for format version to be checked by the parser (currently `UCLA blk 1.0`) followed by the name of the benchmark's author or creating software, along with the date, in free format, e.g.,

```
UCLA blk 1.0 Igor Markov <imarkov@cs.ucla.edu> 02/22/1999
```

The format version is followed by a four-line *header*

```
Regular partitions  : <positive integer>
Pad partitions      : <positive integer>
Relative capacities : <yes/no> [<yes/no>...]
Capacity tolerances : <float><switch> [<float><switch>...]
```

Colons are separated from words and numbers by spaces, and the options are interpreted as follows:

***Relative capacities :*** will partition capacities for all partitions be given in percent ?

`yes` e.g., "90" later in the file would mean 90% of total module area specified in the `.are` file[10]

`no` means that partition capacities are interpreted in absolute units, e.g., area; "90" may then be interpreted as "90 area units".

***Capacity tolerances :*** must be specified for all partitions

`switch` can be empty, "**%**" or "**b**"; "**%**" and "**b**" cannot be separated from the preceding floating point number by spaces

- `%` means that tolerance is interpreted in percent of total module area specified in the `.are` file or by default unit areas
- `b` gives tolerance in multiples of the biggest module area

Both capacities and tolerances can be repeated on the same line to achieve "*multi-capacities*". The number of "multi-capacities" in a given file (*multiplicity*) is determined by the number of *Relative capacities :* specifications. The number of *Capacity Tolerances :* specifications must match, as must the number of capacities of each non-pad partition (see below). Note that this requires "multi-area" files "`.areM`" with matching multiplicity.

The header is followed by an indicated number of regular partition and pad partition specifications; both types of partitions are numbered starting with zero.[11]

[9] Can be used for a brief description of the benchmark, its origin and peculiar features

[10] If no `.are` file is given, all areas are assumed 1.0

[11] This is in slight disagreement with the existing **.netD/.are** file formats, where pads are numbered starting with 1 and other modules from 0. Therefore, pad 1 will often be located in pad partition 0.

Identifiers of regular partitions consist of a "**b**" ("**p**" is taken by `.netD` files, so we think of partitions as "blocks" with "bounding boxes") and their ordinal numbers. Similarly, identifiers of pad partitions are prefaced with "**pb**" ("pad blocks").

It is recommended that partitions be listed in ascending order of their numbers with pad partitions first.

Each "partition line" has the following structure (below "**999**" stands for positive integer)

```
<Partition Id> <Geom Type> <Geom Desc> : [<Capacities>]
```

`Partition Id` can be of the form **b999** or **pb999**

`Geom Type` can be "**rect**" (for rectangle) with additional types possible[12]

`Geom Desc` must be **Xmin Ymin Xmax Ymax** for rectangles. A point is specified by **Xmin = Xmax, Ymin = Ymax**.

`Capacities` for pad partitions will be ignored by the parser. Regular partitions must have exactly as many [multi]-capacities as specified in the file header (see **Relative capacities :** ) and will be interpreted as absolute numbers or relative to the total module area. Relative specifications are given in percent (50 for 50%) and must sum up to 100% or more.

### UCLA `.fix` File Format

Empty lines and lines starting with pound sign '#' are ignored by the parser. The first non-ignored line is reserved for format version to be read by the parser (currently fixed at `UCLA fix 1.0`) followed by the name of the benchmark's author or creating software, along with the date, in free format, e.g.,

```
UCLA fix 1.0 Igor Markov <imarkov@cs.ucla.edu> 02/22/1999
```

The format version is followed by a four-line *header*

```
Regular Partitions : <positive integer>
Pad Partitions     : <positive integer>
Fixed Pads         : <positive integer>
Fixed NonPads      : <positive integer>
```

Colons are separated from words and numbers with spaces. Fixed pads and modules follow in indicated quantities, specified one per line in the form

```
<Module/Pad Id> : <Partition Id> [<Partition Id>...]
```

`Module/Pad Id` can be of the form **a999** or **p999** ("999" means "a positive integer") and must refer to a module or pad (terminal) declared in the `.netD` file

`Partition Id` can be of the form **b999** or **pb999** and, if the geometry of partitions is important, will refer to a partition declared in a .blk file. If no .blk file is given, the parser can use the biggest partition id as a guess for the number of partitions of each type, OR can be given these data by the user.

There are no restrictions on fixing modules and pads in partitions (e.g., one can fix a module in a pad partition or vice versa, a pad can be fixed in several regular partitions as a result of terminal propagation, etc.), and fixing a module/pad in several partitions should be interpreted with an "OR" (i.e., the module can go to any of the indicated partitions). In particular, fixing a module/pad in all regular partitions results in a *free* module/pad and must have the same effect as not mentioning the pad/module in the **.fix** file at all.

[12] We have considered introducing type **point**, for example, but currently find it more convenient to represent points with rectangles.

### References


[1] C. J. Alpert, "Partitioning Benchmarks for VLSI CAD Community", Web page, `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), pp. 1-81.

[5] F. Brglez, "Design of Experiments to Evaluate CAD Algorithms: Which Improvements Are Due to Improved Heuristic and Which are Merely Due to Chance?", *technical report* CBL-04-Brglez, NCSU Collaborative Benchmarking Laboratory, April 1998.

[6] 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* 45(3) (1998), pp. 580-589.

[7] 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.

[8] 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.

[9] 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.

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

[11] 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.

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

[13] 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. Additional publications and benchmark results for hMetis-1.5.3 are available at `http://www-users.cs.umn.edu/~karypis/metis/hmetis/main.html`.

[14] G. Karypis, R. Aggarwal, V. Kumar and S. Shekhar, "Multilevel Hypergraph Partitioning: Applications in VLSI Domain", *technical report*, University of Minnesota Computer Science Department, March 27, 1998.

[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", *Proc. IEEE Intl. Conference on Computer-Aided Design*, 1998, pp. 203-211.

| Circuit | 2% Tolerance | | | 10% Tolerance | | |
|---|---|---|---|---|---|---|
| | 1 Start | 2 Starts | 4 Starts | 1 Start | 2 Starts | 4 Starts |
| IBM01A_L0H | 351/391.7( 9.0) | 348/389.0(11.8) | 330/364.4(17.9) | 318/360.6( 8.1) | 334/362.3(11.9) | 324/349.2(16.2) |
| IBM01A_L0V | 249/293.1( 8.6) | 249/280.6(11.8) | 249/277.2(18.0) | 251/277.6( 8.8) | 248/273.1(10.8) | 249/269.8(17.6) |
| IBM01B_L1_V0H | 227/243.2( 2.7) | 224/232.9( 4.1) | 224/228.9( 6.7) | 227/232.6( 2.5) | 215/225.0( 4.1) | 215/223.3( 5.8) |
| IBM01B_L1_V0V | 286/290.9( 2.6) | 255/290.7( 3.7) | 247/283.6( 6.3) | 245/270.0( 2.6) | 235/264.8( 3.8) | 230/255.3( 5.7) |
| IBM01C_L1_V1H | 289/339.3( 3.1) | 285/333.9( 4.9) | 289/325.0( 8.3) | 276/307.7( 3.3) | 274/314.4( 4.9) | 276/295.7( 7.5) |
| IBM01C_L1_V1V | 360/367.3( 2.9) | 328/355.4( 4.5) | 317/344.3( 7.3) | 338/361.6( 3.1) | 250/275.2( 4.2) | 250/259.5( 7.0) |
| IBM01D_L3_C11-33H | 356/414.4( 4.1) | 392/413.3( 5.0) | 369/406.2( 7.7) | 304/327.4( 3.9) | 304/324.4( 4.8) | 307/316.6( 7.9) |
| IBM01D_L3_C11-33V | 393/402.7( 3.6) | 394/407.2( 4.6) | 394/405.4( 7.1) | 384/399.6( 2.9) | 382/396.5( 4.3) | 382/390.8( 7.2) |
| IBM06A_L0H | 627/728.4(44.8) | 607/723.6(51.4) | 566/674.4(74.4) | 433/459.7(38.1) | 432/454.5(45.8) | 434/443.6(61.9) |
| IBM06A_L0V | 611/763.8(41.9) | 586/713.8(48.8) | 572/638.3(66.8) | 422/456.7(36.5) | 423/440.3(43.0) | 427/435.9(61.1) |
| IBM06B_L1_V0H | 571/575.8( 7.1) | 571/571.5(10.6) | 571/571.2(17.1) | 571/576.4( 7.3) | 571/571.9(10.2) | 571/571.5(16.0) |
| IBM06B_L1_V0V | 363/366.4( 7.7) | 360/365.4(10.1) | 363/365.4(15.6) | 311/329.6( 7.6) | 311/326.6(10.9) | 310/328.0(16.0) |
| IBM06C_L1_V1H | 744/780.6(20.8) | 744/778.0(27.5) | 740/768.8(37.8) | 678/690.2(18.2) | 677/683.8(19.3) | 677/682.8(30.6) |
| IBM06C_L1_V1V | 1797/1797.0( 8.4) | 1797/1797.0(12.4) | 1797/1797.0(18.5) | 1797/1797.0( 8.7) | 1797/1797.0(12.5) | 1797/1797.0(17.9) |
| IBM06D_L3_C11-33H | 743/748.9( 5.1) | 740/744.2( 8.6) | 740/743.2(13.9) | 743/753.2( 5.1) | 740/751.6( 8.3) | 740/745.4(14.0) |
| IBM06D_L3_C11-33V | 1162/1212.3(12.2) | 1125/1146.0(10.8) | 1126/1150.6(17.1) | 1137/1258.3(11.2) | 1104/1157.7(12.3) | 1106/1139.7(16.9) |
| IBM09A_L0H | 583/657.7(83.7) | 586/610.9(97.8) | 583/595.4(118.5) | 481/652.0(74.5) | 449/591.4(95.7) | 462/555.6(111.9) |
| IBM09A_L0V | 586/695.1(77.1) | 583/618.8(96.8) | 588/599.7(112.8) | 472/625.1(76.1) | 476/607.0(105.7) | 466/579.2(110.1) |
| IBM09B_L1_V0H | 20982/21032.5(47.5) | 20981/21030.0(53.4) | 20974/20998.7(62.3) | 20977/20982.0(40.7) | 20973/20980.8(50.9) | 20972/20980.4(58.4) |
| IBM09B_L1_V0V | 26770/27036.4(49.5) | 26559/26832.2(71.9) | 26552/26780.4(92.8) | 19451/19809.5(58.3) | 19219/19561.9(81.2) | 19077/19468.6(101.2) |
| IBM09C_L1_V1H | 19213/19235.6(64.3) | 19213/19243.1(72.9) | 19212/19232.0(94.6) | 19203/19209.9(64.7) | 19202/19217.9(66.5) | 19202/19212.2(98.3) |
| IBM09C_L1_V1V | 40038/40038.0(16.5) | 40038/40038.0(23.2) | 40038/40038.0(32.2) | 33444/33610.0(65.5) | 33096/33307.7(84.8) | 33110/33244.4(108.2) |
| IBM09D_L3_C11-33H | 19387/19393.3(35.8) | 19387/19393.3(43.2) | 19387/19391.8(60.6) | 19385/19390.9(27.6) | 19387/19390.7(35.6) | 19387/19390.0(50.8) |
| IBM09D_L3_C11-33V | 20258/20273.5(46.9) | 20262/20276.5(56.8) | 20264/20272.8(82.7) | 20258/20260.1(25.4) | 20258/20260.4(36.8) | 20257/20260.3(56.0) |
| IBM10A_L0H | 1304/1548.2(122.8) | 1344/1468.9(139.6) | 1368/1453.5(205.7) | 1079/1177.8(106.7) | 1058/1119.3(134.5) | 1064/1085.1(161.1) |
| IBM10A_L0V | 1294/1557.0(117.7) | 1373/1524.6(160.4) | 1292/1465.8(210.4) | 1004/1163.5(81.0) | 1001/1140.8(114.6) | 1001/1113.7(177.3) |
| IBM10B_L1_V0H | 1041/1109.2(23.2) | 1004/1092.2(27.2) | 980/1057.4(44.0) | 903/959.7(17.6) | 902/948.3(26.7) | 878/931.0(35.6) |
| IBM10B_L1_V0V | 935/1012.2(16.0) | 935/1011.2(23.7) | 935/995.8(36.5) | 694/733.8(17.7) | 694/719.0(27.9) | 694/702.5(37.5) |
| IBM10C_L1_V1H | 1274/1604.2(60.8) | 1272/1400.8(76.8) | 1273/1313.6(94.4) | 1418/1551.6(62.7) | 1255/1449.2(86.8) | 1255/1430.5(110.3) |
| IBM10C_L1_V1V | 2583/2605.2(69.0) | 2486/2597.8(81.3) | 2580/2597.1(113.1) | 1418/1551.6(62.7) | 1852/2580.8(80.0) | 2140/2587.6(110.8) |
| IBM10D_L3_C11-33H | 1913/1974.9(27.4) | 1898/1966.4(38.6) | 1907/1964.8(52.2) | 1785/1796.7(31.1) | 1786/1792.5(38.1) | 1786/1791.0(55.3) |
| IBM10D_L3_C11-33V | 1966/1975.4(40.3) | 1966/1975.2(40.6) | 1967/1974.3(53.3) | 1966/1975.1(38.4) | 1966/1975.5(38.5) | 1966/1971.4(58.3) |
| IBM11A_L0H | 916/1074.1(93.0) | 899/1005.7(129.6) | 827/938.6(158.1) | 795/922.7(91.6) | 797/860.7(111.3) | 795/851.0(160.2) |
| IBM11A_L0V | 973/1096.0(97.1) | 953/1066.0(115.8) | 916/1062.6(143.6) | 878/1012.6(92.0) | 849/977.4(115.0) | 846/929.0(165.0) |
| IBM11B_L1_V0H | 659/677.6(20.1) | 659/670.2(30.2) | 657/663.9(42.3) | 635/662.7(19.9) | 636/662.1(29.5) | 636/650.0(49.4) |
| IBM11B_L1_V0V | 1325/1403.0(38.7) | 1292/1357.7(47.1) | 1282/1352.6(62.1) | 1211/1272.2(33.0) | 1196/1246.7(39.5) | 1135/1225.1(54.2) |
| IBM11C_L1_V1H | 790/793.5(26.2) | 781/789.6(32.8) | 781/786.8(58.6) | 789/792.4(24.8) | 1196/1246.7(39.5) | 781/789.2(53.8) |
| IBM11C_L1_V1V | 2206/2398.6(41.4) | 2234/2260.8(45.8) | 2213/2247.4(69.1) | 1824/2185.0(40.3) | 1805/2060.9(54.8) | 1764/2000.2(71.9) |
| IBM11D_L3_C11-33H | 1528/1611.4(26.6) | 1526/1570.6(37.3) | 1524/1561.1(53.7) | 1418/1456.3(23.8) | 1419/1456.9(26.2) | 1418/1440.8(45.6) |
| IBM11D_L3_C11-33V | 1842/1934.3(34.7) | 1805/1858.8(39.0) | 1807/1842.0(54.2) | 1801/1854.7(24.5) | 1795/1828.7(36.4) | 1797/1822.5(53.8) |
| IBM12A_L0H | 2279/2634.7(184.6) | 2246/2429.3(219.3) | 2295/2410.8(291.4) | 2385/2578.9(186.0) | 2337/2524.6(176.7) | 2280/2455.6(266.1) |
| IBM12A_L0V | 2298/2652.4(189.1) | 2321/2441.8(227.0) | 2318/2441.5(274.1) | 2169/2561.6(163.2) | 2240/2524.2(187.6) | 2238/2470.4(271.9) |
| IBM12B_L1_V0H | 1681/1740.6(71.4) | 1692/1756.0(81.7) | 1693/1730.5(99.4) | 1510/1518.5(57.2) | 1510/1519.0(64.4) | 1459/1514.8(87.7) |
| IBM12B_L1_V0V | 2063/2424.8(76.7) | 2049/2458.3(85.8) | 2079/2304.8(131.5) | 1904/2264.7(73.3) | 1957/2157.8(90.8) | 1954/2093.3(112.5) |
| IBM12C_L1_V1H | 1581/1633.2(36.2) | 1570/1602.1(46.2) | 1569/1578.4(72.3) | 1444/1472.3(36.4) | 1439/1470.9(41.3) | 1403/1468.6(63.6) |
| IBM12C_L1_V1V | 2194/2357.2(64.2) | 2179/2317.2(79.0) | 2178/2312.4(95.9) | 2029/2203.2(67.2) | 2019/2172.0(71.7) | 1994/2075.6(102.2) |
| IBM12D_L3_C11-33H | 2477/2791.1(37.5) | 2478/2753.0(49.0) | 2474/2755.0(65.2) | 2635/2767.4(30.3) | 2635/2717.2(39.3) | 2640/2721.5(55.1) |
| IBM12D_L3_C11-33V | 2018/2179.2(37.5) | 2017/2092.5(45.3) | 2014/2085.1(64.1) | 2013/2145.8(30.6) | 2013/2145.2(41.7) | 2015/2064.2(57.3) |
| IBM13A_L0H | 1119/1271.0(178.3) | 1090/1215.2(195.0) | 1106/1172.2(244.8) | 1100/1205.7(188.1) | 1070/1168.4(186.6) | 1080/1139.2(244.1) |
| IBM13A_L0V | 1025/1168.8(163.4) | 1002/1143.6(184.7) | 996/1085.0(254.5) | 977/1132.4(161.3) | 982/1108.8(194.0) | 977/1074.0(234.4) |
| IBM13B_L1_V0H | 860/966.7(48.9) | 860/917.4(54.2) | 860/883.4(71.7) | 880/953.9(43.2) | 860/910.0(51.4) | 860/888.6(70.2) |
| IBM13B_L1_V0V | 1292/1445.4(63.0) | 1254/1363.4(68.5) | 1271/1318.3(96.0) | 1098/1215.8(58.9) | 1121/1179.0(73.8) | 1129/1170.7(95.0) |
| IBM13C_L1_V1H | 1106/1139.2(52.4) | 1098/1126.2(63.3) | 1090/1112.9(91.6) | 996/1034.6(72.1) | 995/1022.4(84.5) | 998/1014.6(109.8) |
| IBM13C_L1_V1V | 1249/1317.9(68.2) | 1262/1319.7(77.9) | 1240/1297.6(109.9) | 1176/1217.7(63.2) | 1173/1209.8(73.8) | 1166/1199.2(109.7) |
| IBM13D_L3_C11-33H | 1602/1649.2(58.6) | 1572/1623.6(72.7) | 1576/1610.5(85.5) | 1286/1340.2(53.8) | 1284/1309.1(61.8) | 1284/1303.8(77.6) |
| IBM13D_L3_C11-33V | 1776/1834.0(44.4) | 1771/1812.4(56.6) | 1754/1786.4(78.4) | 1696/1739.3(37.2) | 1698/1718.6(49.4) | 1698/1712.4(72.0) |
| IBM16A_L0H | 1797/2388.8(565.4) | 1767/1973.0(584.8) | 1766/1874.0(761.6) | 1780/2044.8(387.6) | 1765/1890.2(527.6) | 1766/1816.2(667.3) |
| IBM16A_L0V | 1882/2465.7(532.0) | 1879/1959.0(581.1) | 1875/1911.0(772.1) | 1889/2184.5(375.5) | 1847/1944.9(507.6) | 1859/1908.8(748.2) |
| IBM16B_L1_V0H | 1234/1328.5(169.2) | 1222/1280.8(211.1) | 1195/1248.0(308.1) | 1116/1202.3(132.2) | 1124/1172.2(191.0) | 1124/1143.2(261.9) |
| IBM16B_L1_V0V | 2615/2692.9(190.7) | 2609/2633.3(261.0) | 2609/2623.8(314.5) | 2484/2577.7(176.4) | 2473/2535.6(245.5) | 2456/2504.6(310.0) |
| IBM16C_L1_V1H | 1832/1996.2(133.3) | 1828/1998.6(168.4) | 1823/1949.3(261.4) | 1837/1897.7(143.6) | 1848/1896.0(170.5) | 1842/1868.4(245.7) |
| IBM16C_L1_V1V | 2987/3098.5(187.2) | 2783/3041.6(253.4) | 2848/3027.5(320.7) | 2598/2808.2(191.0) | 2560/2746.0(203.4) | 2562/2649.5(291.6) |
| IBM16D_L3_C11-33H | 2516/2732.1(126.0) | 2514/2629.9(162.8) | 2529/2575.5(213.6) | 2317/2413.9(96.0) | 2315/2370.6(141.1) | 2301/2359.0(190.7) |
| IBM16D_L3_C11-33V | 2207/2466.4(99.3) | 2210/2244.6(148.1) | 2214/2243.8(211.8) | 2220/2472.1(114.4) | 2220/2319.7(161.4) | 2212/2267.0(199.2) |
| IBM17A_L0H | 2688/2795.2(570.7) | 2650/2698.3(625.3) | 2531/2646.8(925.7) | 2328/2608.0(515.9) | 2285/2568.2(715.5) | 2281/2454.0(818.6) |
| IBM17A_L0V | 2682/2815.8(544.0) | 2638/2712.9(704.0) | 2632/2733.2(825.9) | 2429/2695.2(636.1) | 2303/2604.9(799.6) | 2390/2538.6(1027.7) |
| IBM17B_L1_V0H | 1888/1902.4(306.1) | 1893/1903.9(315.1) | 1893/1901.7(398.2) | 1891/1909.4(279.6) | 1889/1907.6(333.5) | 1892/1904.3(399.2) |
| IBM17B_L1_V0V | 2799/3021.5(286.1) | 2732/2939.2(344.1) | 2634/2836.8(444.5) | 2122/2421.7(277.6) | 1864/2136.0(313.0) | 1902/1972.6(439.3) |
| IBM17C_L1_V1H | 1713/1738.4(150.1) | 1707/1729.1(183.0) | 1713/1716.3(265.3) | 1559/1626.7(151.1) | 1561/1623.3(200.5) | 1561/1600.4(264.5) |
| IBM17C_L1_V1V | 4000/4120.7(244.8) | 3464/3902.1(246.3) | 3457/3663.3(363.1) | 3147/3486.0(210.4) | 3166/3261.2(250.1) | 3050/3208.6(305.7) |
| IBM17D_L3_C11-33H | 3009/3100.5(97.9) | 2911/3039.5(104.1) | 2922/2987.5(160.1) | 2919/3037.3(88.1) | 2918/2992.4(101.8) | 2913/2953.5(142.5) |
| IBM17D_L3_C11-33V | 3210/3443.6(114.9) | 3287/3337.4(129.2) | 3179/3283.1(175.9) | 3176/3313.0(92.7) | 3173/3292.4(119.1) | 3177/3256.2(149.3) |

Table 3: Results of the hMetis-1.5.3 program on our new benchmarks, with 2% and 10% balance tolerance. Each entry reports min cut / ave cut (CPU sec).