Optimality, Scalability and Stability Study of Partitioning and Placement Algorithms

Jason Cong, Michail Romesis, Min Xie

Computer Science Department

University of California at Los Angeles

Los Angeles, CA 90095

{ cong, michail, xie } @cs.ucla.edu

Abstract

This paper studies the optimality, scalability and stability of state-of-the-art partitioning and placement algorithms. We present algorithms to construct two classes of benchmarks, one for partitioning and the other for placement, which have known upper bounds of their optimal solutions, and can match any given net distribution vector. Using these partitioning and placement benchmarks, we studied the optimality of state-of-the-art algorithms by comparing their solutions with the upper bounds of the optimal solutions, and their scalability and stability by varying the sizes and characteristics of the benchmarks. The conclusions from this study are: 1) State-of-the-art, multilevel two way partitioning algorithms scale very well and are able to find solutions very close to the upper bounds of the optimal solutions of our benchmarks. This suggests that existing circuit partitioning techniques are fairly mature. There is not much room for improvement for cutsize minimization for problems of the current sizes. Multiway partitioning algorithms, on the other hand, do not perform that well. Their results can be up to 18% worse than our estimated upper bounds. 2) The state-of-the-art placement algorithms produce significantly inferior results compared with the estimated optimal solutions. There is still significant room for improvement in circuit placement. 3) Existing placement algorithms are not stable. Their effectiveness varies considerably depending on the characteristics of the benchmarks. New hybrid techniques are probably needed for future generation placement engines that are more scalable and stable.

Categories and Subject Descriptors

B.7.2 [**Hardware**]: INTEGRATED CIRCUIT--*Design Aids*

General Terms

Algorithms, Design, Performance

Keywords

Optimality, Scalability, Stability, Partitioning, Placement

1. INTRODUCTION

The circuit partitioning problem has been studied since the 1970's. The Kernighan-Lin (KL) algorithm [1] was introduced in 1970, and later extended to the Fiduccia-Mattheyses (FM) algorithm [2] in 1982. Following that, there was a period of 15 years in which little progress was made on partitioning. Significant progress was made during the mid-to-late-90's [3, 4, 5, 6, 7, 8, 9, 10, 11]. In less than five years, the best reported cutsize on commonly used benchmarks was reduced by almost 50%. For example, Fig. 1 shows the progress of bipartitioning algorithms through the years for the MCNC and ISPD98 benchmarks, where we use FM as a comparison basis. PANZA [11] was introduced in 1995 and is a hybrid of eigenvector optimization method and the recursive max-flow min-cut method. CLIP (1996) [5] and LSR (1997) [7] are using iterative improvement techniques, while hMetis [3] is based on the multilevel framework. After the introduction of hMetis no significant improvement was reported, raising the question of whether we have reached a plateau.
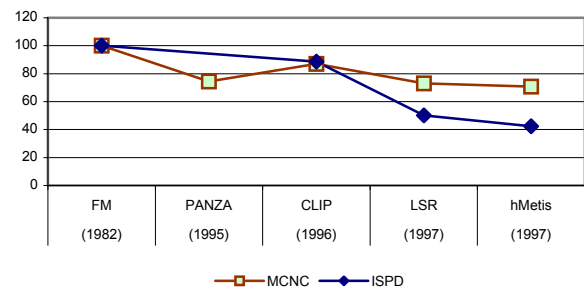

Figure 1. Quality Improvement of Bipartitioning Algorithms over the Years for the MCNC and ISPD98 Benchmarks.

Contrary to partitioning, the development in placement has been slow but steady. Various algorithms have been proposed in the past 30 years, including min-cut methods, iterative methods, and analytical methods. In terms of wirelength reduction, the rate of improvement has been only 5-10% every 2-3 years since the 1980s. In 1988 Gordian [12] reported substantial wirelength reduction over its predecessors. In 1991 Gordian-L [13] reported a 20% wirelength reduction over Gordian. TimberWolf v.7.0 [14] reduced Gordian's wirelength by 10% in 1993. The iterative force-directed method [15] outperformed Gordian-L in 1998 by an average of 6%. The mPL placer [16] runs 10x faster than Gordian-L with a penalty of wirelength increase of 10%. The latest developments in placement algorithms in the past three years, including Capo [17], Dragon [18], Mongrel [19], and mPG [20]

vary mostly in runtime. The wirelength difference between Dragon and Capo is within 5%, but Dragon is 7x slower [21]. mPG is about 2x faster than Dragon with an up to 5% longer wirelength [22]. Mongrel's wirelength is also slightly worse than Dragon's [23]. This lack of significant progress also prompts us to wonder whether there remains much room for improvement in circuit placement (at least in terms of wirelength minimization).

Note that in all these studies, the evaluation of an algorithm is based on comparison with other existing algorithms; there is little understanding about how far away the solutions are from the optimal. There were several efforts on the optimality study of VLSI CAD algorithms in the past. Synthetic benchmarks with known optimal solutions were created to study the partitioning problem [24, 25]. However, they used highly regular structures and may not provide reliable performance estimates for general cases. A noticeable effort was [26], which quantified the sub-optimality of several VLSI CAD heuristics by stitching smaller designs into larger ones. Such construction bounded the optimal solution of large designs to be at most the best achievable solution to the original design multiplied by the scaling factor. Using these examples, [26] studied several then state-of-the-art partitioning (FM [2], MBC-FM [5], EIG1 spectral [22] and Rcut1.0 ratio cut [23]) and placement (TimberWolf, Gordian-L) algorithms. It concluded that significant improvement was possible over these algorithms. Very recently, an algorithm was developed in [29] to construct placement examples with known optimal wirelength, by using a method by Boese[1] as described in [26]. Surprisingly the results showed that the wirelength produced by three recently-published placers, Capo v.8.0, Dragon v.2.20, and mPL v.1.2, ranges from 1.46 to 2.38 times the optimal value on those examples.

However, both the studies in [26] and [29] have limitations. Since [26] was published seven years ago, it did not evaluate the latest algorithms, such as hMetis, MLPart [30], ESC/LSR [31], etc., for partitioning, and Capo, Dragon, mPG, etc., for placement. Moreover, compared with the design complexity of today, the test cases used in [26] are quite small (the largest netlist is about 40K). It is necessary to extend the optimality study to the new algorithms in the era of multi-million gate designs. The study in [29] had only local nets in the optimal solutions, which may not be true for real circuits. Furthermore it did not perform stability studies of existing placement algorithms with similar sizes, yet different characteristics.

This paper studies the optimality, scalability and stability of state-of-the-art partitioning and placement algorithms using arbitrarily large, specially constructed benchmarks, which can match any net degree distribution with a known bound of the optimal solution. We study the optimality of these algorithms by comparing their solutions with the bounds, their scalability and stability by varying the sizes and characteristic of the benchmarks. Benchmark suites for hypergraph bipartitioning (BEKU – Bipartitioning Examples With Known Upper Bound) and multiway partitioning (MEKU-Multiway Partitioning Examples with Known Upper Bound) that match the net degree distribution of the ISPD98 benchmarks [33] are constructed. The performance of three state-of-the-art partitioning algorithms, hMetis [3], MLPart [30] and Flare [31], is evaluated on these benchmarks. We also derive two benchmark

suites for placement. One is called G-PEKU (Placement Examples with Known Upper bound and Global connections only). The other suite, PEKU (Placement Examples with Known Upper bound), is derived by introducing non-local connections into the PEKO suite [29], which consists of local connections only. Four state-of-the-art placement algorithms, Capo v.8.5 [17], Dragon v.2.20 [18], mPG v.1.0 [20] and mPL v.1.2 [16] are evaluated on these benchmarks.

The rest of this paper is organized as follows. Section 2 describes the study of partitioning algorithms. Section 3 describes the study of placement algorithms. In Section 4 we give the conclusions.

2. STUDIES OF PARTITIONING ALGORITHMS

2.1 Construction of Bipartitioning Examples with Known Upper Bounds

In this section we present an algorithm for the construction of bipartitioning examples with known upper bound of their optimal cutsize. First, we introduce some notations: Given a netlist N, its Net Distribution Vector (NDV) is defined as a vector $D = (d_2, d_3,...,d_k)$, where d_m represents the number of nets of degree m in N.

Given a netlist N of size n, an NDV $D = (d_2, d_3,...,d_k)$ and a cut size bound B, we want to construct a circuit N' such that:

- N' has n nodes and it has D as its NDV.

- The minimum balanced bipartitioning cutsize of N' is at most B.

The description of the algorithm is as follows: The nodes are first divided into two partitions of equal size. The nets are constructed as follows: Let α be equal to $B / \sum_{i=2}^{k} d_i$. For every integer i from 2 to k, we create $\lfloor \alpha \cdot d_i \rfloor$ nets of degree i that include nodes from both partitions and $d_i - \lfloor \alpha \cdot d_i \rfloor$ nets of degree i that include nodes from only one partition.   It is not difficult to see that the cutsize of this partitioning solution is less than or equal to B. The two partitions are then fed as an initial solution to FM [2], which tries to improve the cutsize by moving cells across the partition. Although the initial partitions are of equal size, FM is allowed to create unbalanced partitions up to a limit. In our bipartitioning experiments, the unbalance factor is set to be 5% (i.e., the largest partition can occupy up to 55% of the total area). The final result provided by FM is an upper bound of the optimal cutsize.

We call the benchmarks that are created by this algorithm BEKU (Bipartitioning Examples with Known Upper Bound). Our motivation behind the creation of these benchmarks was to create circuits with known partitioning solutions by construction which, although not optimal, are better than the solutions that state-of-the-art algorithms can find. Although this algorithm does not guarantee the latter part, our experiments showed that for a range of the value of B (usually between 20-27% of the total number of nets), the solutions by our construction are almost[2] always better than the solutions produced by hMETIS, MLPart and Flare (see Table 1).

---

[1] Although K. Boese was the first who proposed this method, he did not carry it to implementation to construct these examples [32].

[2] In 2 out of 8 examples, hMetis reported a slightly better cutsize than the estimated upper bound (the difference on average for these two cases was only 0.01%)

2.2 Construction of Multiway Partitioning Examples with Known Upper Bounds

The algorithm for the creation of the BEKU benchmarks can be easily extended to create benchmarks for multiway partitioning. In this section we present the algorithm for the creation of Multiway partitioning Examples with Known Upper bound (MEKU).

Given a netlist N of size n, an NDV $D = (d_2, d_3, ..., d_k)$, a cut size bound B and a number of partitions m, we want to construct a circuit N' such that:

- N' has n nodes and it has D as its NDV.

- The minimum balanced m-way partitioning cutsize of N' is at most B.

Similar to the previous algorithm, the nodes are first divided into m partitions of equal size. The nets are constructed as follows: Let α be again equal to $B / \sum_{i=2}^{k} d_i$. For every integer i from 2 to k, we create $\lfloor \alpha \cdot d_i \rfloor$ nets of degree i that include nodes from at least two partitions and $d_i - \lfloor \alpha \cdot d_i \rfloor$ nets of degree i that include nodes from only one partition. The cutsize of this partitioning is less than or equal to B. For the solution refinement we apply the multiway variation of FM [34].

2.3 Evaluation of State-of-the-art Partitioning Algorithms on the BEKU and MEKU Suites

We created a total of 16 benchmarks that are available for download from the following web address:

http://cadlab.cs.ucla.edu/~pubbench/partitioning/download. Eight of them are BEKU examples and the rest are MEKU examples for 8-way partitioning. The benchmarks match the average NDV of the ISPD98 benchmarks and their sizes range from 0.5M to 2M nodes. For the BEKU set we tried two different values for the bound B: 20% and 25% of the total number of nets. For the MEKU set we also tried two values for the bound B: 30% and 35% of the total number of nets. We evaluated three state-of-the-art partitioning algorithms:

- hMetis: hMetis runs on a multilevel framework. It consists of three phases: hypergraph coarsening, initial partitioning and refinement. It uses the MHEC and FC clustering algorithms during coarsening and FM during refinement. The version we used to run our experiments is hMetis v1.5 for Linux. For the parameters of the program, we used the values suggested in the hMetis manual [35]. For the multiway partitioning examples, we used the recursive bipartitioning version.

- MLPart: MLPart is also multilevel-based and uses different algorithms for coarsening (PinEC) and refinement (VRW). The version we used is MLPart v4.19.1 for Linux. MLPart does not support multiway partitioning, therefore the reported results are restricted to the BEKU suite.

- Flare: Flare is a multiway partitioning algorithm based on the PM [36] multiway partitioning framework and the LR [7] bipartitioning engine on top of a two-level hierarchy created by the ESC [31] clustering algorithm. We tested the version available internally at the UCLA VLSI CAD Lab.

All the experiments were performed with a Pentium 4 2.2GHz machine running RedHat Linux 8.0 with 1.5GB of memory. All programs were run 20 times and we report the best results. For the bipartitioning examples the size of a partition can range from 45-

55% of the total area, while for the multiway examples each partition can occupy 11.8-13.3% of the total area.

Figure 2 shows the experimental results on the BEKU suite and Figure 3 the results on the MEKU suite. The figures show the average ratio of the reported cutsize to the upper bound of the optimal cutsize for every partitioning algorithm. We call this ratio, the Quality Ratio (QR) of an algorithm throughout the paper. Experiments showed that the improved upper bound after the application of FM or multiway FM is, on average, 9% better than the initial upper bound B. For a complete table of all the results, refer to the following web site: http://cadlab.cs.ucla.edu/~pubbench/partitioning.

From the experimental results, we can draw the following conclusions:

- For our BEKU examples, MLPart produces the best results, while Flare produces the worst results. MLPart can also consistently find our estimated solution even for large size circuits. The result seems to suggest that existing circuit bipartitioning techniques are fairly mature and there is not much room for improvement in cutsize minimization.

- For the MEKU examples on 8-way partitioning, hMetis produces better results than Flare, but can still be 18% away from the upper bounds in some cases. This suggests that there is room for improvement for multiway partitioning.

- The value of B influences the quality of hMetis and Flare. In the case of the BEKU examples, for a low value of B, the algorithms report the same results as our upper bound, while for higher values, the results are 25% away from the upper bound. MLPart consistently reports the same results as our upper bound, independently of the value of B.
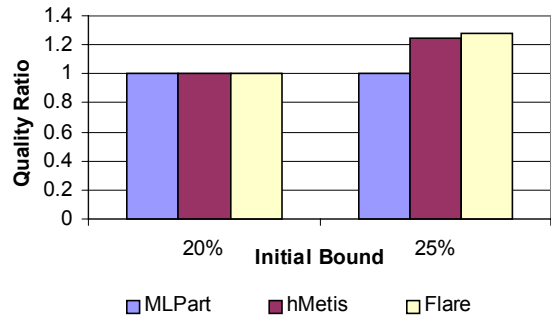

Figure 2. Partitioning results for the BEKU suite. MLPart consistently reports the same cutsize as our estimated upper bound.
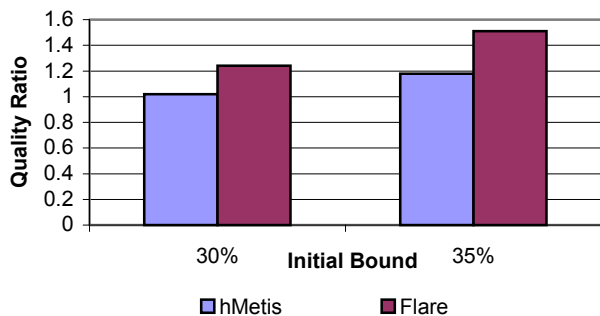
Figure 3. Partitioning results for the MEKU suite. hMetis is worse only by 2% when the initial bound is 30%, but the gap increases to 18% for a bound of 35%.
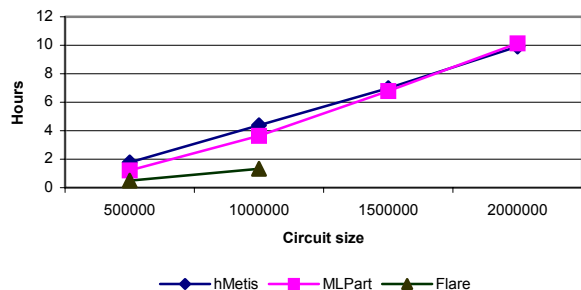

Figure 4. Runtime comparison of hMetis, MLPart and Flare (20 runs total)

The algorithms seem to scale well. Fig. 4 shows the runtime comparison of the three algorithms for the bipartitioning examples with the value of *B* equal to 25% of the nets. Flare is the fastest program, but its memory requirements make it impossible to run the experiments with more than 1M nodes.

3. STUDIES OF PLACEMENT ALGORITHMS

3.1 Optimality and Scalability Study of Existing Placement Algorithms

First, we briefly review the recent optimality and scalability study in [29]. An algorithm is given in [29] that takes a number *n* and a vector *D* as input, and outputs a placement benchmark which has *n* modules and *D* as its NDV with a known optimal half perimeter wirelength. The benchmarks generated are named PEKO, given in both GSRC BookShelf and LEF/DEF formats, and can be downloaded from: http://cadlab.cs.ucla.edu/~pubbench/peko.htm.

Three recently published placers are tested with PEKO, including:

- Capo: Capo is built on a multilevel framework with recursive bisection. It aims to enhance routability with such techniques as tolerance computation and block splitting heuristics [17]. The version used in [29] is Capo v.8.0.

- Dragon: Dragon uses hMetis to derive an initial partition of the netlist. Then it uses simulated annealing with bin-based

swapping to further refine the result. The version used in the experiment is Dragon v.2.20.

- mPL: mPL [16] is a multilevel placement algorithm. It uses nonlinear programming to handle the non-overlapping constraints on the coarsest level, then uses Goto-based [38] relaxation in subsequent refinement stages. The version used is mPL v.1.2. It has an additional V cycle with distance-based clustering in the second V cycle [39].

The conclusions from [29] include:

- There is still significant room for improvement in placement algorithms. The results by these tools are 1.46 to 2.38 times the optimal on these examples.

- The solution quality of all three tools further deteriorates as the circuit size increases. This suggests that these algorithms may not be effective in searching the solution space when the problem size becomes considerably large.

- Capo shows the best scalability in these examples, while Dragon consumes more than 24 hours in cases of design complexity higher than 450K cells.

3.2 Construction of Placement Examples with Known Upper Bounds

The study with the PEKO suite suffers from one drawback. In the optimal solutions, all the nets are local, i.e., their wirelength is the minimum possible value. This may not be true in real circuits, which may also have global connections that span the entire chip, even when they are optimally placed. Table 1 gives the profile of placed results of the ISPD98 suite [33] produced by Dragon. The second and third columns are the width and height of the chip. The fourth column gives the wirelength of the longest net in each circuit. The last column gives the percentage of wirelength contributed by the longest 10% of the total nets. It can be seen that although the number of global connections is small, their wirelength contribution is significant. Therefore, the performance of a placer can be quite different due to the presence of these global nets. It is worthwhile to study the stability of different placers in the presence of global nets.

We take two approaches to consider the impact of global nets. One is to generate circuits consisting only of global nets; the other is to introduce some randomly generated, non-local nets into the PEKO suite. We use the term "non-local net" to denote the nets in placement solutions whose wirelength are larger than the minimum possible value. These nets are used to mimic the global nets.

3.2.1 Placement examples with global connections only

One way to study the impact of global connections is to create circuits with global nets only. Our construction procedure takes an integer *n*, and $0 \leq \alpha \leq 1$ as inputs. It outputs a netlist *N'* and a placement solution *S*, such that *N'* has *n* modules and *S* has an aspect ratio of α. Each net in *N'* connects either an entire row or column, as shown in Fig. 5. The number of nets is the total number of rows and columns. Such an example is similar to datapath placement examples, where data flows horizontally through the bit-slice along buses, while control signal flows vertically. One solution to such benchmarks has a configuration similar to Fig. 5, whose wirelength is the sum of the length of each

row and column, which is obviously an upper bound of the optimal wirelength.

Table 1. Profile of placement result by Dragon [18] on ISPD98

| circuit	height	width	WL of longest net	WL contribution of longest 10%
ibm01	8158	4530	7148	51%
ibm02	8158	6430	14224	46%
ibm03	8158	6740	10624	58%
ibm04	8158	9140	15171	53%
ibm05	8158	11055	19064	47%
ibm06	8158	8715	13966	61%
ibm07	8158	14605	14051	51%
ibm08	8158	15895	16142	60%
ibm09	8158	16395	13780	55%
ibm10	8158	27890	30755	53%
ibm11	16350	10925	19234	59%
ibm12	16350	15545	26748	52%
ibm13	16350	12230	19539	59%
ibm14	16350	25475	26370	61%
ibm15	16350	23785	27284	63%
ibm16	16350	34015	42860	59%
ibm17	16283	38895	45686	56%
ibm18	16350	37065	52846	64%
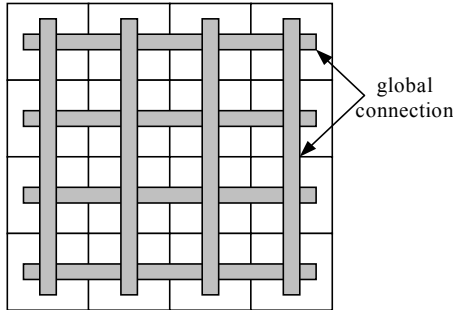
Figure 5. Circuit with global connections only

3.2.2 Placement examples with non-local connections

The second approach is to introduce non-local nets into benchmarks which have only local nets in the optimal solution. Compared with local nets, the non-local nets usually have a longer wirelength, and are used to mimic the global connections in our study. We would like to solve the following problem: Given an integer n, a vector D, and $0 \le \alpha \le 1$, construct a new netlist N' and a placement solution S, such that:

- N' has n modules and has D as its NDV

- The ratio of non-local nets to the total number of nets in S is α

We extended the algorithm in [29] by relaxing the optimality requirement for a subset of the nets. The new algorithm works as follows: The modules are put in a $\lceil \sqrt{n} \rceil \times \lceil n / \lceil \sqrt{n} \rceil \rceil$ shaped region.

For nets of degree i, $\alpha \cdot d_i$ of them are generated by randomly connecting i modules. The rest are generated by connecting i modules in a rectangular region with dimension $\lceil \sqrt{i} \rceil \times \lceil i / \lceil \sqrt{i} \rceil \rceil$, as in [29]. Although the optimal wirelength for the generated circuit is no longer known, we can calculate the wirelength of the random nets and add it to the optimal wirelength of the local nets. The sum serves as an upper bound of the optimal wirelength.

3.3 Evaluation of State-of-the-art Placement Algorithms on the G-PEKU and PEKU Suites

Using the module numbers extracted from ISPD98 and an aspect ratio of 1, we generated a set of circuits with global nets only. The circuits are named GPeku01 to GPeku18. They are grouped as the G-PEKU suite. We also generated several sets of benchmarks with non-local nets. The parameter α is gradually increased from 0.25% to 10%. The module numbers and NDVs were extracted from ISPD98. These benchmarks are named the PEKU suite (Placement Example with Known Upper bound). All the circuits used in the experiment can be downloaded from: http://cadlab.cs.ucla.edu/~pubbench/peku.htm.

The same version of Dragon and mPL as in 3.1 were used in our experiments. As for Capo, we used an updated version, Capo v.8.5 for Linux, provided by its authors, downloaded from [37]. We also added another placer:

- mPG: mPG is built on a multilevel framework. It uses FC clustering and hierarchical density control to minimize the overflow of each placement bin during the refinement process. If necessary, it builds an incremental A-tree to optimize routability. We used mPG v.1.0 given in [20].

The results for Capo were collected on a Pentium 4 2.2GHz running RedHat 8.0 with 1.5GB of memory. The results for the other three placers were collected on a Sun Blade workstation 750MHz running SunOS 5.8 with 4GB of memory.

First, we tested the four placers on five circuits in G-PEKU. The experimental results are given in Table 2. The results are the average of 5 runs for each placer. The upper bound of each circuit is given in column "UB." The last four columns give the ratio of a placement's wirelength to the upper bound. mPL exceeds the memory capacity for two of the examples. For those it completed, mPL gives the closest solution to the upper bound. For these examples with all global nets, the gap between their solutions and the upper bound varies between 41% and 102% in the worst case, similar to the results obtained on PEKO [29], which has local nets only. This is another validation that there is significant room for improvement for the placement problem.

Table 2. Evaluation results on G-PEKU

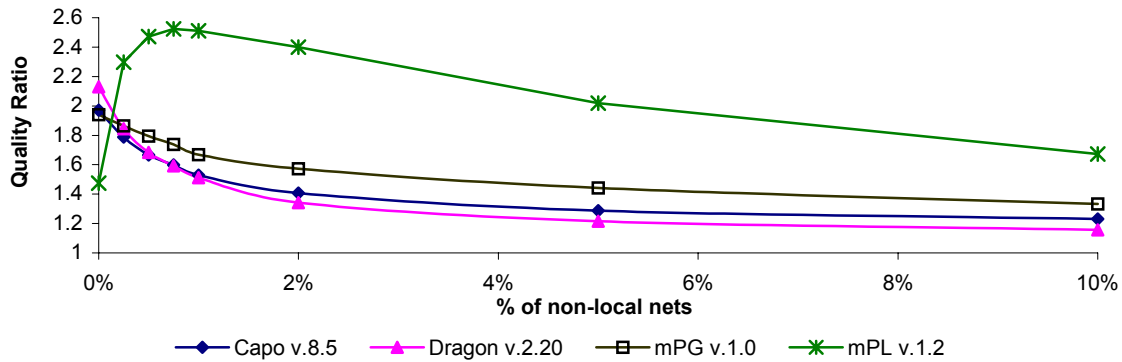| circuit	#cells	UB	Dragon v.2.20 [18] QR	Capo v.8.5 [17] QR	mPG v.1.0 [20] QR	mPL v.1.2 [16] QR
GPeku01	12506	7.93E5	1.98	1.56	1.91	1.11
GPeku05	28146	1.79E6	2.01	1.69	1.97	1.16
GPeku10	68685	4.38E6	2.02	1.72	1.98	1.41
GPeku15	161187	1.03E7	1.99	1.79	1.97	Abort
GPeku18	210341	1.34E7	2.02	1.78	1.98	Abort

Figure 6. QR vs % of non-local nets for four placement algorithms under evaluation

Second, we tested the placers on PEKU. For each α, we picked five of the circuits and fed them into the placers. Each circuit was placed 5 times by the placers. We calculated the QRs as the ratio of the wirelength to the upper bound. Fig. 6 shows the change of the average QRs with the increase of α on PEKU for the four placers. For mPL, its QR increases when α is increased from 0 to 0.75%. For the other three placers, QRs are steadily decreasing. However, this does not necessarily indicate that their solution qualities are improving. We believe it is because the upper bounds of the optimal wirelength are becoming looser as the percentage of non-local nets increases. Therefore, the absolute value of the QRs may not be meaningful. However, comparing QRs from different placers can help us identify the technique that works best under each scenario. Furthermore, comparing the QRs of the same placer can test a placer's sensitivity to non-local connections. It can be seen that the relative effectiveness of the placers changes as the number of non-local nets increases. For instance, the solution quality of Dragon is not close to the optimal when the benchmarks are dominated by local nets. However, it gradually stands out as the percentage of non-local nets increases. When α reaches 10%, Dragon's results become the closest to the upper bound.

Combining the results from 3.1 and 3.3, we can make the following observations:

- The placers are not stable in the presence of global nets. Without global nets, mPL gives the shortest wirelength. However, its Quality Ratio shows an increase of more than 80% with a small increase of non-local nets. With non-local nets, Dragon's wirelength gradually becomes the closest to the upper bound. A placer's Quality Ratio can vary significantly for designs of similar sizes but different characteristics.

- The study suggests that new hybrid techniques, which are more scalable and stable, may be needed for future generations.

4. CONCLUSIONS

In this paper we studied the optimality, scalability and stability of state-of-the-art partitioning and placement algorithms using arbitrarily large, carefully constructed benchmarks that match the characteristics of real circuits. We studied the optimality of these algorithms by comparing their solutions with the upper bounds of the optimal solutions, their scalability by varying the sizes and characteristics of the benchmarks. The conclusions from this study include:

- The best available bipartitioning algorithms perform and scale very well on the BEKU suite. The result seems to suggest that existing circuit partitioning techniques are fairly mature and there is little room for improvement for cutsize minimization for problems of the current sizes.

- The best available multiway partitioning algorithms do not perform that well. There were examples where the upper bound we computed was 18% better than the results from hMetis.

- The state-of-the-art placement algorithms produce significantly inferior results compared with the estimated optimal solutions. There is still significant room for improvement in circuit placement.

- Existing placement algorithms are not stable. Their effectiveness varies depending on the characteristics of the benchmarks. New hybrid techniques may be needed for future generation placement engines that are more scalable and stable.

6. REFERENCES

[1] B. Kernighan and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs," *Bell Systems Technical Journal*, 49(2), pp. 291-308, 1970.

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

[3] G. Karypis, B. Aggarwal, V. Kumar and S. Shekhar, "Multi-level hypergraph partitioning: Application in VLSI domain," *IEEE Trans. VLSI Syst*, vol. 7, pp. 69-79, 1999.

[4] S. Dutt and W. Deng. "A Probability-Based Approach to VLSI Circuit Partitioning," *Proc. Design Automation Conference*, pp. 100-105, 1996.

[5] S. Dutt and W. Deng, "VLSI Circuit Partitioning by Cluster-Removal Using Iterative Improvement Techniques," *Proc. Physical Design Workshop*, 1996.

[6] B. Riess, K. Doll, and F. M. Johannes. "Partitioning Very Large Circuits Using Analytical Placement Techniques". *Proc. Design Automation Conference*, pp. 646–651, 1994.

[7] J. Cong, P. Li, S. K. Lim, T. Shibuya, and D. Xu, "Large Scale Circuit Partitioning With Loose/Stable Net Removal and Signal Flow Based Clustering," *Proc. International Conference on Computer Aided Design*, pp. 441-446, 1997

[8] J. Cong and M. Smith, "A Parallel Bottom-up Clustering Algorithm with Applications to Circuit Partitioning in VLSI Design," *Proc. Design Automation Conference*, pp. 750-760, 1993.

[9] D. Huang and A. Kahng, "When Clusters Meet Partitions: New Density-Based Methods for Circuit Decomposition," *Proc. European Design and Test Conference*, pp. 60-64, 1995.

[10] H. Shin and C. Kim, "A Simple Yet Effective Technique for Partitioning," *IEEE Trans. On VLSI Systems*, pp. 380-386, 1993.

[11] J. Li, J. Lillis, and C.K. Cheng, "Linear Decomposition Algorithms for VLSI Design Applications," *Proc. Design Automation Conference*, pp. 223-228, 1995.

[12] J. M. Kleinhans, G. Sigl, and F. M. Johannes, "GORDIAN: A New Global Optimization / Rectangle Dissection Method for Cell Placement," *Proc. International Conference on Computer-Aided Design*, pp. 506-509, 1988.

[13] G. Sigl, K. Doll, and F. Johannes, "Analytical Placement: A linear or Quadratic Objective Function?" *Proc. Design Automation Conference*, pp. 427-432, 1991.

[14] W. Sun and C. Sechen, "Efficient and Effective Placement for Very Large Circuits," *Proc. International Conference on Computer-Aided Design*, pp. 170-177, 1993.

[15] H. Eisenmann and F. M. Johannes, "Generic Global Placement and Floorplanning," *Proc. Design Automation Conference*, pp. 269–274. 1998.

[16] T. Chan, J. Cong, T. Kong, and J. Shinnerl, "Multilevel Optimization for Large-Scale Circuit Placement," *Proc. IEEE International Conference on Computer Aided Design*, pp. 171-176, 2000.

[17] A. Caldwell, A. Kahng and I. Markov, "Can Recursiv Bisection Produce Routable Placements?" *Proc. Design Automation Conference*, pp. 477-482, 2000.

[18] M. Wang, X. Yang and M. Sarrafzadeh, "Dragon2000: Standard-cell Placement Tool for Large Industry Circuits," *Proc. International Conference on Computer-Aided Design*, pp. 260-264, 2000.

[19] S. Hur and J. Lillis, "Mongrel: Hybrid Techniques for Standard Cell Placement," *Proc. International Conference on Computer Aided Design*, pp. 165-170, 2000.

[20] X. Yang, B. Choi, and M. Sarrafzadeh, "Routability Driven White Space Allocation for Fixed-Die Standard-Cell Placement," *Proc. International Symposium on Physical Design*, pp. 42-49, 2002

[21] C-C. Chang, J. Cong, Z. Pan, X. Yuan, "Physical Hierarchy Generation with Routing Congestion Control," *Proc. International Symposium on Physical Design*, pp. 36-41, 2002.

[22] X. Yuan, *Personal communication*, Oct. 2002.

[23] J. Lillis, *Personal communication*, Dec. 2001

[24] T. Bui, S. Chaudhuri, F. Leighton, and M. Sipser, "Graph Bisection Algorithms with Good Average Case Behavior," *Combinatorica*, 7(2), pp. 841-855, 1987.

[25] J. Garbers, H. Promel, and A. Steger, "Finding Clusters in VLSI circuits", *Proc. International Conference on Computer Aided Design*, pp. 520-523, 1990

[26] L. Hagen, D. Huang and A. Kahng, "Quantified Suboptimality of VLSI Layout Heuristics," *Proc. Design Automation Conference*, pp. 216-221, 1995.

[27] L. Hagen and A. Kahng, "New Spectral Methods for Ratio Cut Partitioning and Clustering," *IEEE Trans. On CAD* 11(9), pp. 1074-1085, 1992.

[28] Y.C. Wei and C.K. Cheng, "Towards Efficient Hierarchical Designs by Ratio Cut Partitioning," *Proc. International Conference on Computer Aided Design*, pp. 298-301, 1989.

[29] C-C. Chang, J. Cong, and M. Xie, "Optimality and Scalability of Existing Placement Algorithms," *Proc. Asia and South Pacific Design Automation Conference*, pp. 621-627, 2003.

[30] A. Caldwell, A. Kahng, and I. Markov, "Improved Algorithms for Hypergraph Bipartitioning," *Proc. Asia and South Pacific Design Automation Conference*, pp. 661-666, 2000.

[31] J. Cong and S. Lim, "Edge Separability Based Circuit Clustering With Application to Circuit Partitioning," *Proc. Asia South Pacific Design Automation Conference*, pp. 429-434, 2000.

[32] K. Boese, *Personal communication*, Oct. 2002

[33] C. Alpert, "The ISPD98 Circuit Benchmark Suite," *Proc. International Symposium on Physical Design*, pp. 85-90, 1998.

[34] L. Sanchis, "Multiple-way Network Partitioning," *IEEE Trans. on Computers*, pp. 62-81, Vol. 38, Issue 1, 1989.

[35] http://www-users.cs.umn.edu/~karypis/metis/hmetis/files/manual.pdf

[36] J. Cong, and S. Lim, "Multiway Partitioning with Pairwise Movement," *Proc. IEEE International Conference on Computer Aided Design*, pp. 512-516, 1998.

[37] http://www.eecs.umich.edu/~imarkov/4/JRS/

[38] S.Goto, "An efficient algorithm for the two dimensional placement problem in electrical circuit layout," *IEEE Trans. on Circuit and Systems*, vol 28, pp. 12-18, 1981.

[39] N.K. Sze, Personal communication, 2002