

# Global Objectives for Standard Cell Placement

Mehmet Can Yildiz   Patrick H. Madden  
State University of New York at Binghamton  
Computer Science Department  
pmadden@cs.binghamton.edu   http://vlsicad.cs.binghamton.edu

## ABSTRACT

Recursive bisection based placement is well known, and recent advances in partitioning have made the approach more attractive. While partitioners can optimize a placement from a *local* perspective, high performance design requires consideration of *global* issues as well. We focus on aspects of the placement problem which cannot be captured with bisection, addressing them through a new approach derived from recent work on  $k$ -way partitioning. We consider large values of  $k$ , and objective functions which are more complex than the traditional *min-cut*.

Our placement tool, *Feng Shui*, integrates this new  $k$ -way partitioning method into a traditional recursive bisection framework. Experimental results show the effect of the approach; there is reduced variation in solution quality, in 8 of 11 benchmarks best case wire length is improved, and for 9 of 11 benchmarks, average wire length is improved. These improvements are obtained with negligible impact to total run time.

## 1. INTRODUCTION

Circuit placement has been extensively studied; objectives such as area minimization, wire length minimization, and timing optimization are common. In this paper, we focus on standard cell placement problems: our objective is to place rectilinear circuit elements (cells) into one or more horizontal rows, minimizing total wire length. In particular, we are interested in capturing *global* aspects of the problem, improving a traditional approach.

Placement is a difficult problem. For all but trivial problem sizes, we have only heuristic methods; optimization is usually performed on only a small subset of the circuit at any given time. If we perturb a subset of circuit elements, we can optimize a placement from a *local* perspective, but have no guarantee that our modifications are appropriate from a *global* perspective. This distinction is a primary focus of this paper. Our contribution is a method which allows at least some global issues to be captured within a bisection based framework, resulting in improved placement quality.

We adapt a recently presented partitioning approach for  $k$ -way partitioning. Rather than direct  $k$ -way partitioning, we are interested in  $k$  *simultaneous* bisections, for large values of  $k$ . We integrate this

into a traditional bisection based placement framework, resulting in a practical standard cell placement tool we call *Feng Shui*.

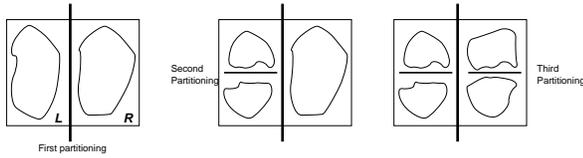
## 2. FORMULATION AND PREVIOUS WORK

We follow traditional problem formulations, and use hypergraphs to model circuit netlists. Cells are denoted as  $c_i$ , and are roughly equivalent to vertices in a hypergraph. Nets are denoted as  $n_j$ , and are roughly equivalent to hyperedges. When placing circuit elements, we will determine *regions* in which the elements lie; these are non-overlapping rectilinear areas, and are denoted by capital letters. Our approach employs partitioning of regions, converting each region into two or more *subregions*.

As the placement problem is well studied, we have a number of established approaches for it. **Force Directed** or **LP** based approaches repeatedly solve systems of equations, determining cell locations iteratively (for example, [5]). This approach is popular in commercial placement tools. **Simulated Annealing** based approaches obtain cell placements by swapping positions of cells randomly, guided by a probabilistic acceptance function. A number of current commercial placement engines utilize this approach; efficient cost estimates allow the consideration of large numbers of intermediate states. A well known example of this approach is TimberWolf [13]. **Partitioning** based approaches determine cell locations by recursively dividing an initial area (region) with successive bisections or quadrisections. This approach has become more attractive recently; advances in partitioning research have provided a number of fast algorithms which produce extremely good results.

Our standard cell placement approach is integrated into a traditional partitioning-based framework. In an early algorithm, Breuer[1] utilized repeated graph bisections to obtain a circuit placement; this approach is shown in Figure 1. The bisections divide the circuit netlist into a hierarchy of cells, with the resulting hierarchy roughly mapping into a rectilinear grid. Dunlop and Kernighan[4] extended this approach, through the use of an improved partitioning method[9], and also *terminal propagation*. When partitioning a region, we can expect a number of connections to be required to cells or pads outside of the region. Terminal propagation provides a simple method to insert fixed “dummy” vertices, so that the partitioning considers these external connections.

Moving beyond simple bisections, Suaris and Kedem[12] explored the use of quadrisection (a four way partitioning). Huang and Kahng[7] also apply quadrisection, utilizing a multi-level clustering based partitioning algorithm, and considering minimum spanning tree lengths, rather than the simple min-cut metric.



**Figure 1: Order of bisections in a top-down recursive approach. The partitioning of  $L$  influences the solution for  $R$ , and vice versa.**

Note that with terminal propagation, the partitionings of regions become interdependent; if we begin with two regions,  $L$  and  $R$ , and partition  $L$  first, this impacts the optimal solution for  $R$ . Partitioning  $R$  first might result in a different solution, and neither of these might be globally optimal, even if the individual partitionings were.

To address the order dependence of the partitioning, both [12] and [7] employ repeated partitioning at each level. We might wish to partition  $L$ , followed by  $R$ , and then partition  $L$  a second time. Repeated partitionings do not, however, change a local optimization process into a global one.

For circuit placement applications, we may have tens or hundreds of thousands of regions, each of which requires partitioning, and each of which has impact on neighboring regions. This problem motivates our interest in  $k$ -way partitioning, or more precisely,  $k$  simultaneous bisections, for extremely large values of  $k$ . Recently, Zhong and Dutt[14] presented a partitioning driven placement approach which also uses large scale multi-way partitioning.

### 3. PLACEMENT APPROACH

Our placement approach integrates a variant of a recent  $k$ -way partitioning approach into a traditional framework. In this section, we first briefly describe the framework, followed by a discussion of the new technique.

#### 3.1 Bisection

The framework for our placement tool might be considered a textbook implementation of the approach of Dunlop and Kernighan[4]. We repeatedly divide the circuit netlist by either horizontal or vertical cut lines. We utilize the recent multi-level clustering based partitioning algorithm hMetis[8], version 1.5.3.

At each partitioning, we attempt to obtain a nearly exact bisection if cutting vertically. If the cut line is horizontal (splitting a number of rows), we split the rows as evenly as possible. If the region being bisected contains an odd number of standard cell rows, we insert fixed and weighted dummy vertices, allowing a nearly exact bisection to be mapped into the space available easily. The partitioning objective is *min-cut*; the hMetis partitioner attempts to minimize the number of cut hyperedges.

When we employ bisection, we may choose to cut a region either horizontally or vertically. In our placement engine, we control the *aspect ratio* of any region, using a parameter  $AR$  to determine the direction of cut lines. If the region being bisected contains more than a single row, we may elect to bisect horizontally if the ratio of the height and width exceeds a user defined threshold.

The recursive bisection process divides placement regions into progressively smaller areas, ultimately assigning each cell to a single

row, but possibly having several cells remaining within a region. To establish positions for each cell, we order them by region location within each row, packing them together without spaces or overlap.

The positions of cells which were within the same region will be arbitrary at this point; they were not ordered by the partitioning process. To optimize these positions, we apply branch-and-bound reordering, modifying the positions of a small set of consecutive cells in a single row.

*Feng Shui* allows the specification of a “window size,” controlling the number of cells involved in any branch-and-bound optimization. This window passes over each cell row (in order), traveling along each row at steps of half the window size. At each step, the optimal order for cells found under the window is determined.

The number of passes over the placement, and the size of the window, are both parameters which can be controlled by the user. In practice, we find that window sizes of 6 to 8 cells, and 4 passes of improvement, are sufficient for good overall performance. Increasing the window size may impact run times substantially (as the complexity of the branch-and-bound procedure is  $O(w!)$  worst case, where  $w$  is the size of the optimization window). In [2], a number of ways to implement branch-and-bound reorderings efficiently were explored.

#### 3.2 $k$ -Way Partitioning

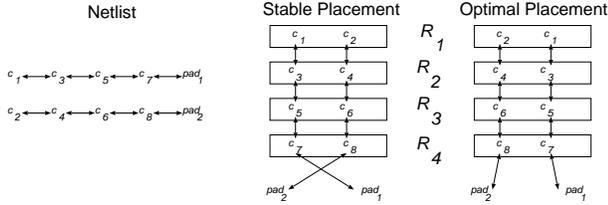
The focus of our work has been on the *global* aspects of the placement problem. With partitioning, we can optimize the number of edges cut within a region effectively, but have no way of knowing if this *local* optimization is appropriate from a *global* perspective. Similarly, our branch-and-bound reordering is also a *local* optimization.

A careful examination of placement by recursive bisection reveals a number of instances where global objectives may be lost. The example in Figure 2 shows a simple case where local optimization is insufficient; we have four regions to bisect, each with two cells. If we approach this problem as a series of independent bisections, a number of configurations which are both stable and suboptimal can be encountered. The suboptimality of the global solution has nothing to do with the quality of the bisections of each region; simply improving the bisection algorithm will not improve the global configuration.

As we progress through the placement process, the number of regions increases, doubling repeatedly. If we have  $k$  regions, and wish to split each of them (obtaining  $2k$  new regions), the traditional approach is iterative, bisecting a single region at a time. Our new approach is to attempt bisection of all regions at the same time, obtaining a solution that is of good quality *globally*. The method used to perform this massive bisection is based on partitioning by iterative deletion[10].

##### 3.2.1 New Formulation

To capture global objectives effectively, we formulate the placement problem as one of variant of *multi-way partitioning*, rather than as a series of bipartitions. We partition all regions *simultaneously*, with the intermediate state of each region influencing the others. We are concerned with partitioning very large numbers of regions, and our cost objective is wire length rather than *min-cut*. The problem we consider is *given a set of regions (with physical constraints) and a set of elements mapped to these regions, bisect*



**Figure 2:** Given the netlist above, we may assign pairs of cells to the regions shown. If we partition or apply branch-and-bound reordering to the four regions, we may be unable to find an optimal solution. If we determine the ordering in region  $R_1$  first, we arrive at a stable and suboptimal solution. Repeated local optimization (through repartitioning) will fail to find the globally optimal solution.

all regions to minimize the resulting bounding-box wire length. Solution of this problem optimizes the circuit from a global perspective.

Multi-way partitioning has proven quite challenging[11]; for traditional objectives such as *min-cut*, the greatest success has been obtained with recursive partitioning, largely ignoring the nature of the problem considered here.

### 3.2.2 Iterative Deletion

In [10], hypergraph partitioning was considered. A new method based on *iterative deletion* was presented; in this approach, vertices are duplicated, with one instance of each vertex being assigned to a partition. Redundant elements are removed one at a time until no duplicates remain.

While the approach was relatively simple, it proved effective in some areas where traditional methods had difficulty. For bipartitioning, cut sizes from a single linear-time pass were comparable to many passes of a traditional FM[6] algorithm. Multi-way cut sizes were superior to a direct flat multi-way partitioning algorithm[11]. For problems with a variety of hyperedge weights, a combination of iterative deletion and FM partitioning proved substantially more effective than FM partitioning alone. The approach is computationally attractive: with integer hyperedge weights, it may be implemented in  $O(n)$  time.

Our variation of the iterative deletion approach for placement operates in the following manner. Each cell in a region is assigned to *both* subregions; if there is more than a single instance of a cell, it is considered to be *redundant*. We repeatedly remove redundant cells from subregions which have high utilization, and select the highest cost cell for removal.

In our current implementation of the placement engine, we evaluate cell cost based on the center of mass for the component nets. For each net  $n_i$ , the center of mass for this net is the average  $X$  and  $Y$  location of the cells which it connects. The cost of any cell  $c_i$  is the sum of the distances between the cell and the center of mass of each net to which the cell is connected.

In this way, a cell which is far from the center of mass of each net to which the cell is connected has high cost. Each region has a number of cells assigned to it, and an available capacity; we remove redundant cells from the region which has the highest ratio of cell

Benchmark	Rows	Cells	Nets	TW Result
fract	6	149	163	0.032
struct	21	1952	1920	0.383
primary1	17	833	904	0.974
primary2	22	3014	3029	3.594
biomed	44	6514	7052	1.690
industry1	24	3085	2594	1.540
industry2	69	12637	13419	14.064
industry3	52	15433	21967	42.264
avqsmall	79	21918	30038	6.152
avqlarge	83	25178	33298	6.580
golem3	117	100312	217362	25.113

**Table 1:** The placement benchmarks considered. The number of rows used was determined by the TimberWolf placement and routing tools.

area to capacity. The cell removed from any region is the redundant cell which has the highest cost.

We utilize heaps to maintain the ordering of cells within any given region, and we also use a heap to maintain an ordering of regions. In this way, maintenance and cell selection are both at worst  $O(\log n)$  for each cell removed. As the number of redundant elements to be removed in any pass is  $n$ , each pass is  $O(n \log n)$ . Region sizes decrease by a factor of 2 with each pass, resulting in a logarithmic number of passes required. Thus, the *iterative deletion* portion of our algorithm is at worst  $O(n \log^2 n)$ .

To illustrate the iterative deletion process, we present Figure 3. In this figure, we duplicate cells  $c_1$ , in region  $R_1$ , and cell  $c_2$  in region  $R_2$ , and assume that a net connects  $c_1$  to  $c_2$ , and that a second net connects  $c_2$  to a pad.

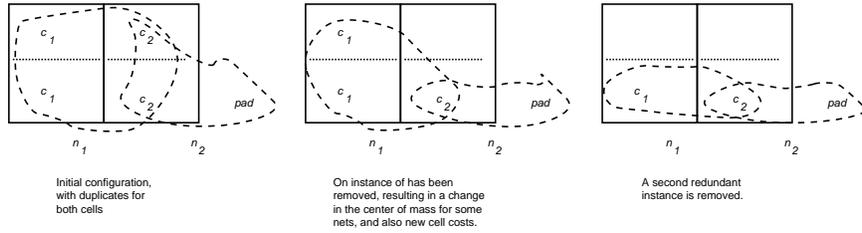
The order of cell deletions in this figure is as follows. Note that other cell deletions may be interspersed with the following; we focus only on these cells to clarify the process.

- The center of mass for the nets connected to cell  $c_2$  is closest to the pad; we remove the instance of  $c_2$  which is furthest from this location (as this is the instance which has highest cost).
- The center of mass for nets connected to cell  $c_2$  is recalculated, and this is propagated to the other cells.
- Net  $n_1$  now has two instances of  $c_1$  and one instance of  $c_2$  connected to it: the center of mass for this net changes, influencing the cost for each instance of  $c_1$ .
- An instance of  $c_1$  is removed.

## 4. EXPERIMENTAL RESULTS

To evaluate the impact of our *global* optimization strategy, we have implemented a standard cell placement engine which utilizes either a traditional Dunlop and Kernighan style recursive bisection approach, or the Dunlop and Kernighan approach combined with iterative deletion. We refer to this tool as *Feng Shui*, and have made it publicly available. All experiments were performed on a 500mhz PentiumIII PC running Linux.

Our expectations were not that we would obtain dramatic reductions in wire length. Global optimization is extremely difficult, and



**Figure 3: Two instances of each cell in the netlist are generated, and assigned to both subregions of any region. Cell costs are based on the center of mass for the vertices; high cost cells are removed one at a time from any region of the entire placement problem. In this way, the partitioning is performed on a global basis, rather than with only a pair of subregions at a time.**

Benchmark	Feng Shui				Feng Shui without Iterative Deletion				Capo			
	Min	Avg	Max	Time(s)	Min	Avg	Max	Time(s)	Min	Avg	Max	Time(s)
fract	0.032	0.033	0.034	2	0.034	0.035	0.036	2	0.035	0.036	0.036	0
struct	0.380	0.389	0.400	46	0.383	0.392	0.405	46	0.380	0.404	0.420	10
primary1	1.018	1.050	1.102	20	1.022	1.056	1.097	20	0.991	1.043	1.078	5
primary2	3.684	3.742	3.842	89	3.668	3.811	4.018	88	3.827	3.902	4.008	26
biomed	1.689	1.715	1.741	179	1.694	1.710	1.742	179	1.789	1.854	1.932	47
industry1	1.606	1.623	1.648	66	1.600	1.617	1.634	66	1.665	1.690	1.721	16
industry2	15.408	15.700	16.027	396	15.331	15.864	16.210	391	15.382	15.626	15.941	130
industry3	44.729	45.484	46.171	592	44.487	46.364	48.262	586	47.575	48.373	49.579	200
avqsmall	5.960	6.063	6.196	1005	5.966	6.126	6.317	988	5.923	6.071	6.289	244
avqlarge	6.301	6.417	6.591	1078	6.291	6.440	6.630	1070	6.207	6.510	6.759	268
golem3	21.882	22.294	23.302	3511	21.816	22.450	23.125	3495	24.581	25.482	26.791	1381

**Table 2: Wire length and run time comparisons of placement using recursive partitioning, and the improvement obtained by considering global objectives. Included also are wire length results for Capo. Feng Shui without iterative deletion is comparable to a Dunlop and Kernighan style approach. Run times are in seconds, and average per run.**

if the work on partitioning was any indication, little progress could be made. Never the less, we feel that a global perspective is crucial to successful high performance design, and wished to determine if any leverage could be obtained by considering the global problem.

Placements were performed on the MCNC standard cell benchmarks, and also the *golem3* benchmark. These are the largest publicly available benchmark circuits. The number of cells, nets, and rows used are shown in Table 1. A commercial version of TimberWolf, 1.2, was used to determine *reference* wire lengths; we note that the TimberWolf results are *optimized for routability, and not with the sole objective of wire length minimization*. As such, these results are not directly comparable to those of *Feng Shui*. All wire lengths are scaled by  $10^6$ .

For comparison with current tools, we also ran *Capo*[3] on the benchmarks. *Capo* has been made publicly available, along with evaluation tools and detailed benchmark information, making independent verification of results possible. *Capo* has been compared favorably to current commercial placement tools, producing low wire length placements that were routable in most experiments with industrial circuits.

As both *Feng Shui* and *Capo* utilize partitioners which use random seeds, we have run *Feng Shui* 20 times and *Capo* 80 times on each benchmark, reporting best, worst, and average results for each. In *Feng Shui*, we apply two iterations of partitioning improvement; the run time is dominated by this activity. *Capo* is from 3 to 4 times faster, and the additional runs allow comparable total run times, but with a greater chance of obtaining a “good” placement result. These results are shown in Table 2; a graphic version of the results

is shown in Figure 4.

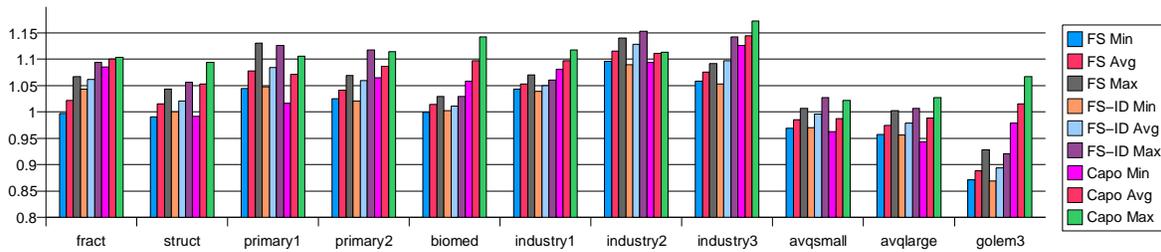
In many benchmarks, the results of *Capo* and *Feng Shui* are similar; as they both utilize recursive bisection and strong partitioning algorithms, this is not surprising.

The performance of *Feng Shui* is extremely good on the largest benchmark, *golem3*, consistently outperforming *Capo* by nearly 11%. The substantial improvement obtained by *Feng Shui* appears to be a result of cut direction selection, and not an artifact of the underlying partitioning algorithms. Methods to optimize cut directions is part of our current work.

When comparing *Feng Shui* with the iterative deletion preprocessing to *Feng Shui* in a basic Dunlop and Kernighan configuration, some benchmarks showed greater improvement than others. This indicates a varying degree of effectiveness of the iterative deletion step. In nine of eleven benchmarks, average wire length is improved, and in eight of eleven benchmarks, best observed wire length is improved. There is reduced variation in results (we note again that the partitioning algorithms utilize random seeds, resulting in different placements with each run). In most cases, consideration of the *global* aspect of the placement problem resulted in improvements in the best, average, and worst case results.

While improvements in general were modest, we make the following observations.

- Consideration of the global nature of the problem can result in improvement; there are issues that cannot be captured, even when we utilize a very effective bisection algorithm.



**Figure 4: Wire lengths, scaled with the TimberWolf result as unity. We evaluate minimum, average, and maximum performance, with Feng Shui without iterative deletion being comparable to a Dunlop and Kernighan style approach.**

- A very simple and efficient method, iterative deletion, can provide a few percent improvement with negligible impact on run time. From this, we assert that we can obtain a modest improvement “for free,” and suspect that improvements in direct  $k$ -way partitioning will have benefit for circuit placement.

## 5. CONCLUSION

In this paper, we have presented an optimization approach which allows the consideration of global objectives from within a traditional top-down placement framework.

Global optimization is extremely difficult; while the conditions under which iterated bisections can obtain stable and suboptimal configurations are clear, it was unknown if this would occur in practice, or if it would be at all common. With a relatively simple approach, some improvement has been obtained, indicating that not only do these configurations occur, but that we can address them through a global approach. While the average improvement is relatively small, these gains cannot be obtained through local optimization alone. We expect that greater improvements are possible, and are investigating methods to obtain improved direct  $k$ -way partitions.

Timing driven placement is a significant concern for modern design. We are currently working with an industry research group to evaluate the performance of our approach on large designs under realistic delay rules. We note that delay optimization is perhaps more of a global phenomena than wire length minimization: meeting timing objectives may require modifications in many areas of a placement, and reductions in delay for some nets may require increased delay in others.

We also observe that the traditional approach of Dunlop and Kernighan is quite effective when modern multi-level clustering partitioners are utilized. Wire lengths were comparable to those of a well known commercial tool, and results reported for the tool *Capo* indicate that placements are not necessarily difficult to route. Our implementation of *Feng Shui* is fast; we find solutions for the largest available benchmark circuits, using modest hardware and CPU times. The nearly linear growth in run times should allow application to extremely large industrial circuits.

## 6. REFERENCES

- [1] M. A. Breuer. A class of min-cut placement algorithms. In *Proc. Design Automation Conf*, pages 284–290, 1997.
- [2] A. E. Caldwell, A. B. Kahng, and I. L. Markov. Optimal partitioners and end-case placers for standard cell layout. In *Proc. Int. Symp. on Physical Design*, pages 90–96, 1999.
- [3] Andrew E. Caldwell, Andrew B. Kahng, and Igor L. Markov. Can recursive bisection alone produce routable placements? In *Proc. Design Automation Conf*, pages 477–482, 2000.
- [4] A. E. Dunlop and B. W. Kernighan. A procedure for placement of standard-cell VLSI circuits. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, CAD-4(1):92–98, January 1985.
- [5] H. Eisenmann and F. M. Johannes. Generic global placement and floorplanning. In *Proc. Design Automation Conf*, pages 269–274, 1998.
- [6] Charles M. Fiduccia and R. M. Mattheyses. A linear-time heuristic for improving network partitions. In *Proceedings of the 19<sup>th</sup> IEEE Design Automation Conference*, pages 175–181, 1982.
- [7] D. J.-H. Huang and A. B. Kahng. Partitioning based standard cell global placement with an exact objective. In *Proc. Int. Symp. on Physical Design*, pages 18–25, 1997.
- [8] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: Application in VLSI domain. In *Proc. Design Automation Conf*, pages 526–529, 1997.
- [9] Brian W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 49:291–307, 1970.
- [10] P. H. Madden. Partitioning by iterative deletion. In *Proc. Int. Symp. on Physical Design*, pages 83–89, 1999.
- [11] L. A. Sanchis. Multiple-way network partitioning with different cost functions. *IEEE Trans. on Computers*, 42(22):1500–1504, 1993.
- [12] P. R. Suaris and G. Kedem. An algorithm for quadrisection and its application to standard cell placement. *IEEE Trans. on Circuits and Systems*, 35(3):394–303, 1988.
- [13] W.-J. Sun and C. Sechen. Efficient and effective placement for very large circuits. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 14(3):349–359, 1995.
- [14] Ke Zhong and Shantanu Dutt. Effective partition-driven placement with simultaneous level processing and global net views. In *Proc. Int. Conf. on Computer Aided Design*, volume 2000, pages 254–259, 2000.