Fixed-outline Floorplanning Through Better Local Search

Saurabh N. Adya and Igor L. Markov
Univ. of Michigan, EECS department, Ann Arbor, MI 48109-2122
{*sadya,imarkov*}*@eecs.umich.edu*

Abstract

Classical floorplanning minimizes a linear combination of area and wirelength. When Simulated Annealing is used, e.g., with the Sequence Pair representation, the typical choice of moves is fairly straightforward.

In this work, we study the fixed-outline floorplan formulation that is more relevant to hierarchical design style and is justified for very large ASICs and SOCs. We empirically show that the fixed-outline floorplan problem instances are significantly harder than the well-researched instances without fixed outline. Furthermore, we suggest new objective functions to drive simulated annealing and new types of moves that better guide local search in the new context. Our empirical evaluation is based on a new floorplanner implementation **Parquet-1** *that can operate in both outline-free and fixed-outline modes.*

Our proposed moves are based on the notion of floorplan slack. The proposed slack computation can be implemented with all existing algorithms to evaluate sequence pairs, of which we use the simplest, yet semantically indistinguishable from the fastest reported ([16]). A similar slack computation is possible with many other floorplan representations. In all cases, the slowdown is by a constant factor — roughly 2x.

1 The Problem

We describe the classical floorplanning framework and contrast it to the modern fixed-outline formulation.

1.1 Classical Outline-Free Floorplanning

A typical floorplanning formulation entails *a collection of "blocks"*, which can represent circuit partitions in applications. Each block is characterized by *area* (typically fixed) and *shape-type*, e.g., fixed rectangle, rectangle with varying aspect ratio, an L-shape, a T-shape, or a more general rectilinear polygon, etc (such shapes may optimize layouts of special types of circuits, e.g., datapaths). A solution to such a problem, i.e., a *floorplan*, specifies a selection of block shapes and overlap-free placements of blocks. Depending on shape constraints, a floorplanning formulation can be discrete or continuous. For example, if at least one block is allowed to assume any rectangular shape with fixed area and aspect ratio in the interval $[a,b]$ (where $a < b$) the solution space is no longer finite or discrete. Multiple aspect ratios can be implied by an IP block available in several shapes as well as by a hierarchical partitioning-driven design flow for ASICs [14, 9] where *only the number of standard cells* in a block (and thus the total area) is known in advance. In many cases, e.g., for row-based ASIC designs, there are only finetly many allowed aspect ratios, but solution spaces containing a continuum are used anyways, primarily because existing computational methods cannot handle such a large discrete solution space directly [9]. We point out that in the classical floorplanning formulations, movable blocks tend to have fixed aspect ratios, but the overall floorplan is not constrained by an outline. While several recent works allow for variable block aspect ratios, the more modern fixed-outline formulation (see below) has not been addressed.

Objective functions not directly related to area typically involve a *hypergraph* that connects given blocks. While more involved hypergraph-based objective functions have been proposed, the popularity of the HPWL (half-perimeter wirelength) function is due to its simplicity and relative accuracy, given that routes are not available. The HPWL objective became even *more relevant* [9] with the wide use of multi-layer over-the-cell routing in which more nets are routed with shortest paths.

A fundamental theorem for many floorplan representations, e.g., [10], says that *at least one area-minimal placement can be represented*. This does not hold for objectives that include wirelength because none of the optimal solutions may be "packed" which implies than more nets can be routed with shortest paths.[1]

For the remaining part of this work, we will be dealing with the area and HPWL objectives only, but even this simplified setting implies *multi-objective optimization*. Mathematically, best trade-offs are captured by *the non-dominated frontier* (NDF). Definition: a solution of a multi-objective optimization problem belongs to *the non-*

[1] A simple example: blocks B_1 and B_2, connected by two 2-pin nets to fixed pins P_1 and P_2. The blocks touch in no optimal solution if the pins are sufficiently far from each other.

dominated frontier iff no other solution improves upon one of the objective functions while preserving (or improving) other objective functions.[2] Of those works on abstract floorplanning that address both objectives, most minimize a linear combination [15, 12, 16] with arbitrarily chosen coefficients. By a simple corollary of the definition of NDF, this produces non-dominated solutions, most likely different for different coefficients. Note, however, that area and wirelength have different dimensions. Given that net lengths have the same order of magnitude as the x and y dimensions of the floorplan itself, areas tend to be several orders of magnitude larger than wirelengths and path delays. Moreover, the difference depends on the number of nets, number and size of blocks, etc. In our experiments, area terms dominated wirelength terms unless highly problem-specific coefficients were used. In other words, **it is difficult to fully automate a floorplanner that explores non-dominated solutions with respect to wirelength and area objectives**.[3]

1.2 Modern Fixed-outline Floorplanning

The two mismatches in classical approaches to floorplanning (troubled multi-objective optimization and representations which may not capture any min-wirelength solutions), are graciously resolved in the context of modern ASIC design. As pointed in [9, 2], modern hierarchical ASIC design flows based on multi-layer over-the-cell routing naturally imply **fixed-die** placement and floorplanning rather than the **variable-die** style, associated with channel routing, two layers of metal and feedthroughs. Each top-down step of such a flow may start with a floorplan of prescribed aspect ratio, and with blocks of bounded (but not fixed) aspect ratios. The objective is to minimize wirelength subject to (i) the fixed floorplan outlines, and, perhaps, additionally (ii) zero dead space. Floorplans with no dead space are called "mosaic" in [6]. The constraint (i) implies that the dead space is no longer an objective, but rather a constraint, because it can be computed in advance. This modern floorplanning formulation was proposed in [9], but has not yet been addressed in any works known to us, partly due to the lack of benchmarks. Our work addresses this formulation.

The modern floorplanning formulation proposed in [9] is an inside-out version of the classical outline-free floorplanning formulation — the aspect ratio of the floorplan is fixed, but the aspect ratios of the blocks can vary. Therefore, it is natural to ask which aspects of classical floorplanning research are relevant to the new formulation. We make the following observations:

1. The new formulation, via the zero dead-space requirement, makes research on classical "block packing" *more relevant*. That is because all wirelength-minimal solutions in this formulation can be captured by compacted representations such as sequence pairs [10], O-trees [12], B^*-trees [3] and corner block lists [6]. In fact, *any* floorplan with zero dead space can be captured by known representatios, because it is "compacted".

2. Multi-objective minimization of area and wirelength, — via linear combinations or otherwise — is no longer an issue since dead-space is fixed.

3. Handling blocks with variable aspect ratios appears increasingly important because there may be very few or no floorplans with a given outline for any given fixed configuration of aspect ratios. A number of works [7, 11, 4, 17] handle the floorplan sizing problem, i.e., changes of aspect ratios without reordering blocks, by methods of mathematical optimization (convex linear and non-linear programming). However, such methods are difficult to combine with combinatorial optimization and entail excessive runtimes, for example [17] cites runtime of 19.5 hours for the ami49 benchmark (other works cite smaller runtime). Additionally, such approaches entail a mix of two very different computational engines. The implementation reported in [6] appears to handle discrete variable aspect ratios by randomized re-instantiation of blocks based on a set of 16 alternatives.

4. Perhaps, the greatest shortcoming of known approaches to floorplanning with respect to the new formulation is the lack of appropriate neighborhood structures, i.e., incremental changes ("moves") that preserve the fixed outline of the floorplan. Notably, every floorplan encoded by the corner block list(CBL) representation [6] has zero dead-space (is "mosaic"), but CBL based moves can change the floorplan's aspect ratio considerably.

5. Given that the new floorplanning formulation is *more constrained*, we see increased relevance of research on accomodating application-specific constraints, such as alignment, abutment, order, regions [16], symmetry [13], etc.

We conclude that classical floorplanning is largely relevant to the new floorplanning formulation proposed in [9], however the new formulation may be addressable through other ways than novel representations. **Existing representations and manipulation algorithms do not allow effective traversals of the solution space without violating specific constraints, including the new fixed-outline floorplan constraint**. Alternatively, temporary violations

[2] The design of optimization heuristics can be viewed as a problem with at least two objective functions — runtime and solution quality [1].

[3] Additionally, when the balancing coefficients are found, the hard-to-capture correlation between wirelength and area appears to create particularly difficult local minima for their linear combination.

could be tolerated or fixed. For example, not every corner block list [6] yields a valid floorplan, but the feasibility constraint is clearly stated in [6] and tolerated by the reported implementation.

In this work, we study neighborhood structures for the well-known sequence pair representation. Our proposed *slack-based moves* are more likey to reduce the floorplan span in a given direction (H or V) than random pair-wise swaps and block rotations used in most works based on sequence pairs.

2 Existing Methods

An overwhelming majority of floorplanners rely on the Simulated Annealing framework [14] but differ by internal floorplan representations.

The sequence pair representation for classical floorplans of N blocks has been proposed in [10]. Unlike most new graph-based representations, it consists of two permutations (orderings) of the N blocks. The two permutations capture geometric relations between each pair of blocks. Recall that since blocks cannot overlap, one of them must be to the left or below from the other, or both. In sequence pair

$$(<\ldots,a,\ldots,b,\ldots>,<\ldots,a,\ldots,b,\ldots>) \Rightarrow a \text{ is to the left of } b \quad (1)$$

$$(<\ldots,a,\ldots,b,\ldots>,<\ldots,b,\ldots,a,\ldots>) \Rightarrow a \text{ is above } b \quad (2)$$

In other words, every two blocks constrain each other in either vertical or horizontal direction. The sequence pair representation is shift-invariant since it only encodes pairwise relative placements. Therefore, placements produced from sequence pairs must be aligned to given horizontal and vertical axes, e.g., $x = 0$ and $y = 0$. Multiple sequence pairs may encode the same block placement, e.g., for three identical square blocks, both $(<a,c,b>,<c,a,b>)$ and $(<a,c,b>,<c,b,a>)$ encode the placement with a straight on top of c and b aligned with c on the right.

The original work on sequence pair [10] proposed an algorithm to compute placements from a sequence pair by constructing the horizontal (H) and vertical (V) constraint graphs. The H and V graphs have $N+2$ vertices each — one for each of N block, plus "the source" and "the sink". For every pair of blocks a and b there is a directed edge $a \rightarrow b$ in the H graph if a is to the left from b according to the sequence pair (Formula 1). Similarly there is a directed edge $a \rightarrow b$ in the V graph if a is above b according to the sequence pair (Formula 2) — exactly one of the two cases must take place. Vertices that do not have outgoing edges are connected to the sink, and vertices that do not have incoming edges are connected to the source. Both graphs are considered vertex-weighted, the weights in the H graph represent horizontal sizes of blocks, and the weights in the V graph represent vertical sizes of blocks. Sources and sinks have zero weights.

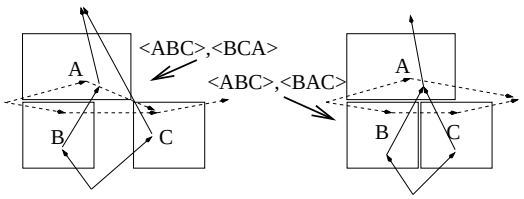

Figure 1. **Two sequence pairs with edges of the horizontal (dashed) and vertical (solid) constraint graphs.**

Block locations are the locations of block's lower left corners. The x locations are computed from the H graph, and y locations are computed from the V graph *independently*. Therefore, we will only look at the computation of the x locations. One starts by assigning location $x = 0$ to the source. Then, the H graph is traversed in a topological order. To find the location of a vertex, one iterates over all incoming edges and maximizes the sum of the source location and source width. Figure 1 illustrates the algorithm on two examples. The worst-case and average-case complexity of this algorithm is $\Theta(n^2)$, since the two graphs, together, have a fixed $\Theta(n^2)$ number of edges, and topological traversals take linear time in the number of edges.

We say that a block placement is "representable" (or "can be captured") by sequence pair, *iff* there exists a sequence pair which encodes that placement. A fundamental theorem from [10] implies that at least one minimal-area placement is representable with sequence pair (in fact, there are many). Therefore, sequence pair is justified for area minimization.

Sequence pairs can be used to floorplan hard rectangular blocks by Simulated Annealing [10, 11, 15, 16]. The moves are (i) random swaps of blocks in one of the two sequence pairs, and (ii) rotations of single blocks. Sequence pairs are modified in constant time, but need to be re-evaluated after each move. No incremental evaluation algorithms have been reported, therefore, the annealer spends most of the time evaluating sequence pairs.

The sequence pair representation and necessary algorithms have been extended to handle fixed blocks [11] as well as arbitrary convex and concave rectilinear blocks [5]. Recently, the original $O(n^2)$-time evaluation algorithm [10], has been simplified and sped up to $O(n\log(n))$ in [15], and then to $O(n\log(\log(n)))$ [16]. Importantly, those algorithms do not change the semantics of evaluation — they only improved runtime, and lead to better solution quality by enabling a larger number of iterations during the same period of time. While O-trees [12] and corner block lists [6] can be evaluated in linear time, the difference in complexity is dwarfed by implementation variations and tuning, e.g., the annealing schedule. The implementation reported in [16] seems to outperform most known implementations, suggesting that the sequence pair is a competitive floorplan representation.

In our experiments, the simple $O(n^2)$ evaluation algorithm from [15] performed faster than the $O(n\log(n))$-time algorithm from the same paper. This is primarily due to the simplicity of data structures used by the $O(n^2)$-time algorithm and their much lower implementation overhead. A more recent paper [16] claims that their advanced $O(n\log(\log(n)))$-time algorithm outperforms the quadratic algorithm in practice. Given that it is considerably more involved, but based on the same principles, we chose to base our work on the quadratic algorithm, leaving out a potential speed up.

All three algorithms are based on the following theorem [15]: The *x*-span of the floorplan to which sequence pair (S_1, S_2) evaluates, equals to the length of the longest common weighted subsequence of S_1 and S_2, where weights are copied from block widths. Analogous statement about the *y*-span deals with the longest common subsequence of S_1^R and S_2 , where ^{R} stands for "reversed" and weights are copied from block heights. Moreover, the computations of *x* and *y* locations of all blocks can be integrated into the longest common subsequence computations.

3 Better Local Search

We propose several ideas for improved move selection in Simulated Annealing and greedy floorplan optimization. While we detail these ideas for the sequence pair representation, they can potentially be applied with other floorplan representations.

3.1 Slack computation

Our first idea can be used with any of above mentioned sequence pair evaluation algorithms and is based on the following series of observations

- *x* and *y* locations are computed independently;
- in each dimension, the floorplan is constrained by one or more "critical paths" in respective constraint graphs. A critical path is a path of blocks that constrain each other in the same direction and are tightly packed so that any change in block location must produce overlaps or increase the span of the floorplan;
- in each dimension, the computation of block locations based on the constraint graphs is mathematically identical to the propagation of arrival times in Static Timing Analysis. Formally, STA is performed on an edge-weighted graph, while the constraint graphs are vertex-weighted, However, this difference is superficial since a vertex-weighted graph can be trivially transformed into an edge-weighted graph, e.g., by distributing vertex weights to incident edges, or otherwise;
- after the *x*-span X of the floorplan and *x*-locations of blocks are known, one can perform a symmetric computation of locations in right-to-left direction, assigning location X to the *sink* vertex. This will be analogous to the back-propagation of required arrival times in Static Timing Analysis;
- by analogy with Static Timing Analysis, the difference between the two locations computed for each block — slack — related to the "most critical path" on which this block lies. In particular, zero slacks are always associated with paths that constrain the floorplan. Negative slacks are impossible as long as blocks do not overlap.

Slacks can be computed with any sequence pair evaluation algorithm that can work in left-to-right and right-to-left modes, which includes all algorithms we are aware of. Figure 2 annotates blocks in a given floorplan with horizontal (*x*) and vertical (*y*) slacks.

3.2 Slack-based moves

Once slacks are known, they can be used in move selection. Both the *timing analysis* interpretation above and the *common subsequence* interpretation from [15] imply that if a move (such as pairwise swap) does not involve at least one block with zero slack in a given dimension, then the floorplan span in that dimension cannot decrease after the move. This is because such a move cannot not improve critical paths or, equivalently, longest common subsequences. Therefore we bias move selection towards blocks having zero slack in at least one dimension. Of those blocks, the ones with large slack in the other direction are potentially good candidates for single-block moves, such as rotations and more gradual aspect ratio changes, — discrete or continuous — can be chosen efficiently. Blocks with two zero slacks, especially small blocks, are good candidates for a new type of move, in which a block is moved simultaneously in both sequence pairs to become a neighbor of another block (in both sequence pairs, and, thus in placement). Namely, we attempt to move a critical block C next to a block L with as large slacks as possible, since large slacks imply that white space can be created around L (more precise conditions can be written, but will still be heuristic). Figure 2 illustrates such a move. In addition to changing the sequence pair, our implementation changes block orientation and aspect ratio based on current slacks.

We observe that [8] already suggested the analogy with static timing analysis in the context of FPGA placement. However, their algorithms are rather different and explicitly rely on H and V constraint graphs (our proposed algorithms do not).

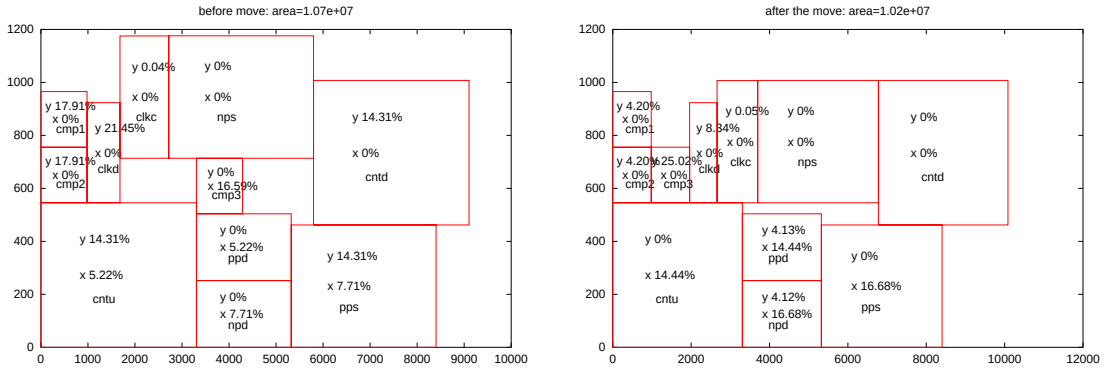

Figure 2. A slack-based move in a highly suboptimal floorplan of hp. x and y slacks are shown in percents of the respective spans of the floorplan. Module `cmp3` — the smallest with zero y slack — is moved next to the module `clkd`, which has the highest y slack. This move improves vertical span and slightly worsens the horizontal span, but the floorplan area is reduced.

3.3 Fixed-outline Constraints

Fixed-outline floorplans enable top-down design of very large scale ASICs and SOCs. Figure 3 shows the result of a floorplan with pure area minimization without any fixed outline constraints. The dead space achieved was 7.75% with an aspect ratio of 3.22:1. However this floorplan can be completely useless for a situation where 1:1 aspect ratio is imposed by a higher-level floorplan.

The following notation will be used in our floorplanning formulations. For a given collection of blocks with total area A and given *maximum percent of dead-space* γ, we construct a fixed outline with aspect ratio $\alpha \geq 1$.[4]

$$H_* = \sqrt{(1+\gamma)A\alpha} \quad W_* = \sqrt{(1+\gamma)A/\alpha}$$

Aside from driving the annealer by area minimization, we consider the following objective functions: (i) the sum of the excessive length and width of the floorplan, (ii) the greater of the two. Denoting the current height and width of the floorplan by H and W, we define these functions as

(i) $\max\{H-H_*,0\}+\max\{W-W_*,0\}$ **(ii)** $\max\{H-H_*,W-W_*\}$

The choice of these functions is explained by the fact that the fixed-outline constraint is satisfied when and only when each of those functions takes value 0.0 or less. For this reason we cannot consider the product of fixed outline violations.

Our experiments have shown that a classic annealer-based floorplanner was practically unable to satisfy the fixed-outline constraint. Therefore we additionally bias the selection of moves as follows. At regular time intervals during the simulated annealing the current aspect ratio is compared to the aspect ratio of the fixed outline. If the two are different, then the slack-based moves described earlier are applied to change the current aspect ratio in the needed direction. For example, if the width needs to be reduced then we chose the blocks in the floorplan with smallest slack in the x direction and insert them above or below the blocks with largest slack in the y direction. These moves have better chances of reducing the area and improving the aspect ratio of the current floorplan at the same time. Through these repeated moves during the simulated annealing the structure of the floorplan is biased towards the aspect ratio of the fixed outline. As shown in the following section, our implementation was successful in satisfying a variety of fixed-outline constraints.

[4]The restriction of $\alpha \geq 1$ is imposed without loss of generality since our floorplanner can change orientations of individual blocks.

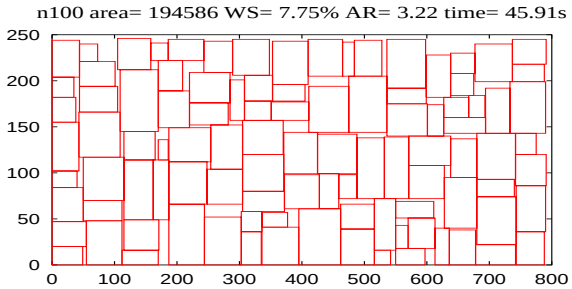

Figure 3. A floorplan with 100 blocks, generated w/o a contraining outline, has aspect ratio 3.22:1.

4 Empirical Validation

We implemented a floorplanner based on Simulated Annealing, **Parquet-1**, that mostly follows a geometric cooling schedule. For a design with N blocks, the temperature is decreased by a factor of α every $1.5N$ blocks. At certain deterministically defined temperatures, α varies. Runtimes are measured (in seconds) on a 800MHz PC/Intel system that runs Linux. Implementations are in C++ and compiled with `g++ 2.95.2 -03`.

Circuit	Enh. O-Tree[12]		TCG[18]		CBL[6]		FastSP[16]		SP(**Parquet-1**)		
	area (mm^2)	time (sec)	area (mm^2)	time (sec)	area (mm^2)	time (sec)	area (mm^2)	time (sec)	min/avg area (mm^2)	min/avg deadspace (%)	avg time (sec)
apte	46.92	11	46.92	1	NA	NA	46.92	1	47.07/48.14	1.08/3.28	4
xerox	20.21	38	19.83	18	20.96	30	19.80	14	19.83/20.73	2.42/6.65	3
hp	9.16	19	8.94	20	NA	NA	8.94	6	9.14/9.49	3.39/6.95	4
ami33	1.24	118	1.20	306	1.20	36	1.20	20	1.19/1.23	2.85/6.01	9
ami49	37.73	406	36.77	434	38.58	65	36.50	31	37.27/38.01	4.91/6.76	16

Table 1. Outline-free area minimization results for Enhanced O-Tree(on Sun Ultra60), TCG(on Sun Ultra60, CBL(on Sun Sparc 20, Fast-SP(on Sun Ultra 1) and Parquet-1(on 800MHz PC/Intel system). Averages and minima for Parquet-1 are over 100 independent starts.

4.1 Classical (min-area) floorplanning context

Table 1 compares **Parquet-1** to leading-edge floorplanning results on standard MCNC benchmarks in the area-only minimization context with no fixed-outline constraints. According to those results, our floorplanner is competitive with published implementations both in terms of final area and runtimes. We note, however, that all recently reported floorplanners easily achieve dead-space well below 10%, therefore leaving very little possible improvement. Simultaneous minimization of wirelength and area is known to be a more challenging optimization problem. In the following, we are going to show that, even without wirelength minimization considered, fixed-outline floorplanning is significantly harder than outline-free floorplanning.

4.2 Fixed-outline floorplanning

The standard version of the floorplanner without any of the slack based moves could not solve a single instance within the fixed outline, although it gave competitive area results. This confirms the inadequacy of the classical min-area floorplanning formulation and algorithms in the fixed-outline context.

To achieve fixed-outline floorplan, we consider three objectives in terms of excessive height and width as described earlier (the sum of and the greater of) and the area. We stop the annealer as soon as it finds a solution satisfying a given fixed outline. If the current outline is smaller, its aspect ratio can be different from the aspect ratio of the fixed outline. If the annealer's temperature schedule runs out and no satisfying solution is found, we deem this a failure.

We constrained our final solutions to have a maximum deadspace of 15% and tried to achieve floorplans satisfying different fixed-outlines. Experiments were performed on **ami49** MCNC benchmark and the results were averaged for 50 runs for each aspect ratio. Figure 4 shows plots of (i) *the probability of success* of satisfying the fixed outline constraint vs desired aspect ratio of the fixed outline, and (ii) *the average runtimes for all runs* vs the desired aspect ratio of fixed outline. The plots reflect the difficulty in satisfying fixed-outline floorplans with given aspect ratios, which highly depends on the dimensions of the blocks. As seen from the plots, our simulated annealer fairly often failed to satisfy the given outline, however, the probability of success was typically over 30%, i.e., at least three in ten starts were successful. This consistent rate of success suggests that our slack-based moves indeed improve local search (simulated annealing without slack-based moves was never able to satisfy the fixed outline). Also note that in most of the unsuccessful attempts the final solutions were within 1-2% from the desired outline, yet *we regard them as failures*.

Out of the three objective functions we tried, minimizing the sum of excessive width and height and minimizing the area was more successful by far. Finding an explanation of this empirical result remains an open problem.

When we decreased γ in our experiments, some fixed outlines were never satisfied, which may be due to the absence of solutions with a given aspect ratio and very small dead-space. As expected, increasing γ improves both the probability of success and the runtime of successful runs. Our experiments with other publicly available benchmarks (n50, n100, etc) produced consistent results.

5 Conclusions

Our work points out that a non-standard floorplanning formulation — fixed-outline floorplanning is significantly harder than classic min-area outline-free floorplanning, even if wirelength minimization is ignored. We implement an annealing-based floorplanner **Parquet-1** that uses a recently discovered [15] sequence pair evaluation algorithm and study its performance both in the fixed-outline and outline-free contexts. For the standard formulation, our floorplanner is competitive, both in terms of runtime and solution quality, with other leading-edge implementations and represents current state-of-the-art. However, our implementation experiences serious difficulties in the fixed-outline context until the algorithm is modified. In particular, more relative deadspace is required to satisfy an outline of a given area when its aspect ratio is fixed.

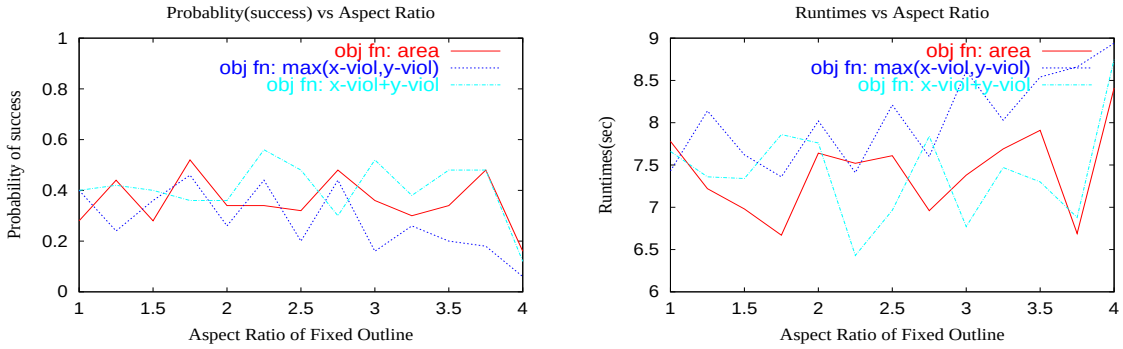

Figure 4. Probabilities of success and average runtimes for fixed-outline floorplan ami49 performed by annealing with three alternative objective functions and slack-based moves. In order to remove noise we plotted average of 50 runs for each aspect ratio.

We propose new objectives that more successfully drive our annealing-based floorplanner to satisfy fixed-outline constraints. We also propose new types of moves that may be applicable to most floorplanner implementations based on simulated annealing, and provide better control of the *x* and *y* dimensions of the floorplan.

Our experiments show that classical methods fail for fixed-outline instances constructed from standard MCNC benchmarks, but when new objectives and slack-based moves are added to our **Parquet-1** implementation, it finds acceptable fixed-outline floorplans for a variety of aspect ratios. We also conclude that minimizing the sum of excessive width and height is a more successful approach than minimizing the greater of the two.

In our on-going research we are extending the proposed methods to fixed-outline floorplanning with wirelength minimization and related applications to standard-cell placement with large macro cells.

References

[1] A. E. Caldwell, A. B. Kahng, A. A. Kennings and I. L. Markov, "Hypergraph Partitioning for VLSI CAD: Methodology for Reporting, and New Results", *DAC '99*, pp. 349-354.

[2] A. E. Caldwell, A. B. Kahng and I. Markov, "Can Recursive Bisection Alone Produce Routable Placements?", *DAC 2000*, pp. 477-482.

[3] Y.-C. Chang, Y.-W. Chang, G.-M. Wu and S.-W. Wu, "B^*-Trees: A New Representation for Non-Slicing Floorplans", *DAC 2000*, pp. 458-463.

[4] P. Chen and E. S. Kuh, "Floorplan Sizing by Linear Programming Approximation", *DAC 2000*, pp. 468-471.

[5] K. Fujuyoshi and H. Murata, "Arbitrary Convex and Concave Rectilinear Block Packing Using Sequence Pair", *ISPD '99*, pp. 103-110.

[6] Xianlong Hong et al., "Corner Block List: An Effective and Efficient Topological Representation of Non-Slicing Floorplan", *ICCAD 2000*, pp. 8-13.

[7] T.-S. Moh, T-S. Chang and S. L. Hakimi, "Globally Optimal Floorplanning for a Layout Problem", *IEEE Trans. on Circuits and Systems - I*, vol 43, no. 9, pp. 713-720, September 1996.

[8] S. Nag and K. Chaudhary, "Post-Placement Residual-Overlap Removal with Minimal Movement", *DATE '99*, pp. 581-586,

[9] A. B. Kahng, "Classical Floorplanning Harmful?", *ISPD 2000*, pp. 207-213.

[10] H. Murata, K. Fujiyoshi, S. Nakatake and Y. Kajitani, "VLSI module placement based on rectangle-packing by the sequence pair", *IEEE Trans. on CAD*, vol 15(12), pp. 1518-1524, 1996

[11] H. Murata and E. S. Kuh, "Sequence-Pair Based Placement Methods for Hard/Soft/Pre-placed Modules", *Proc. ISPD '98*, pp. 167-172.

[12] Y. Pang, C.-K. Cheng and T. Yoshimura, "An Enhanced Perturbing Algorithm for Floorplan Design Using the O-tree Representation", *ISPD 2000*, pp. 168-173.

[13] Y. Pang, F. Balasa, K. Lampaert and C.-K. Chang, "Block Placement with Symmetry Constraint Based on the O-Tree Non-Slicing Representation", *DAC 2000*, pp. 464-468.

[14] N. Sherwani, "Algorithms For VLSI Design Automation", *Kluwer*, 3rd ed. 1999.

[15] X. Tang, R. Tian and and D. F. Wong, "Fast Evaluation of Sequence Pair in Block Placement by Longest Common Subsequence Computation", *DATE 2000*, pp. 106-111.

[16] X. Tang and D. F. Wong, "FAST-SP: A Fast Algorithm for Block Placement Based on Sequence Pair", *ASPDAC 2001*.

[17] F. Y. Young, C. C. N. Chu, W. S. Luk and Y. C. Wong, "Floorplan Area Minimization Using Lagrangian Relaxation", *ISPD 2000*, pp. 174-179.

[18] Jai-Ming Lin and Yao-Wen Chang, "TCG: A Transitive Closure Graph Based Representation for Non-Slicing Floorplans", *DAC 2001*, pp. 764-769.