

Hypercrossing Number: A New and Effective Cost Function for Cell Placement Optimization

Debabrata Ghosh Franc Brglez Matthias Stallmann

December 1998

1998-TR@CBL-12-Ghosh, Version 1.0

Reprinted, with permission, from
<http://www.cbl.ncsu.edu/publications/>

Publications at this site are occasionally revised,
please check for the latest version under the same title.

For more information about CBL, visit
<http://www.cbl.ncsu.edu/>

or write to
info@cbl.ncsu.edu

About This Document

Acknowledgments. F. Brglez and D. Ghosh have been supported by contracts from the Semiconductor Research Corporation (94-DJ-553), SEMATECH (94-DJ-800), and DARPA/ARO (P-3316-EL/DAAH04-94-G-2080 and DAAG55-97-1-0345), and a grant from Semiconductor Research Corporation.

Abstract – *Traditionally, the algorithms that optimize placement of cells in a VLSI layout rely on a cost function that includes a distance metric. The choice of a cost function may limit the choices in devising new, more efficient and more effective algorithms. The metric we propose relies on the interval graph of net nodes, induced by each specific placement of cells into a row. Nominally, the metric associated with this graph is the chromatic number (χ). The new cost function, proposed in this paper, is the hypercrossing number (HC), defined by the cardinality of the edge-set of the interval graph. This metric relates to the edge crossing number (EC) in the corresponding bipartite graph, depends on cell and net order only, and is independent of the target technology.*

Currently, there is no explicit algorithm to place cells such that the hypercrossing number is minimal, but edge crossing minimization algorithms are available. We demonstrate, experimentally and with statistical significance, the following results for the linear placement problem: (1) cost functions based on distance metric and technology-specific wire length measured after routing have a positive linear correlation with the hypercrossing number (with a coefficient of correlation nearly 100%); (2) total wiring area, measured after routing, is highly correlated with the product of the hypercrossing and chromatic number; (3) the edge crossing number in the corresponding bipartite graph model is highly correlated with the hypercrossing number; (4) relatively simple heuristic algorithms that minimize the edge crossing number in bipartite graphs also effectively minimize the hypercrossing number, producing linear placements of quality that can be better than or comparable to state-of-the-art placement algorithms – but at a much lower cost of computation.

Keywords: cell placement optimization, crossing number and wire length minimization.

Note. This document has been published for viewing on the Web in Postscript and HTML formats. The latter is annotated with active hyperlinks to facilitate quick browsing of this and related documents.

Citation. If you choose to cite this report, please add the following entry to your bibliography database:

```
@techreport{
1998-TR@CBL-12-Ghosh,
author = "D. Ghosh and F. Brglez and M. Stallmann",
title = "{ Hypercrossing Number: A New and Effective Cost Function for
          Cell Placement Optimization}",
institution = "{CBL, Computer Science Department, NCSU, Box 7550,
              Raleigh, NC 27695--7550}",
number = "1998-TR@CBL-12-Ghosh",
month = "December",
year = "1998",
note = "{Also available at
        {\tt http://www.cbl.ncsu.edu/publications/}}"}
}
```

Contents

| | | |
|----------|-----------------------------------|-----------|
| 1 | Introduction | 3 |
| 2 | Background and Motivation | 5 |
| 3 | Design of Experiments | 8 |
| 4 | Experimental Results | 11 |
| 5 | Conclusion and Future Work | 18 |

List of Figures

| | | |
|----|--|----|
| 1 | Graph models and key parameters associated with a linear cell placement. | 5 |
| 2 | Summary of experiments with the isomorphism class of the circuit example in Figure 1. | 7 |
| 3 | Design of experiments to compare the placement produced by different algorithms (treatments). | 9 |
| 4 | Workflow for treatments and typical correlations. | 9 |
| 5 | Wire length, as reported by two routers (OASIS and VPR), for two equivalence classes and four placements (Treatments 7, 19, 20, 21). | 11 |
| 6 | Experiments with the <code>ref_multi1_032</code> mutation class and Treatment 19 after routing with OASIS: a correlation report. | 12 |
| 7 | Experiments with the <code>ref_multi1_032</code> mutation class and Treatment 21 after routing with OASIS: a correlation report. | 14 |
| 8 | Experiments on asymptotic behavior with eighteen equivalence classes and three placements (Treatments 19, 20 and 21): reporting wire length after OASIS routing. | 15 |
| 9 | Experiments on asymptotic behavior with eighteen equivalence classes and three placements (Treatments 19, 20 and 21): reporting area after OASIS routing. | 16 |
| 10 | Experiments on asymptotic behavior with ten equivalence classes and two placements (Treatments 19 and 21): reporting wire length after VPR routing. | 17 |
| 11 | Experiments on asymptotic behavior with ten equivalence classes and two placements (Treatments 19 and 21): reporting average CPU time for the placement. | 17 |

Hypercrossing Number: A New and Effective Cost Function for Cell Placement Optimization

Debabrata Ghosh Franc Brglez Matthias Stallmann

CBL (Collaborative Benchmarking Lab), Dept. of Comp. Science, Box 7550, NC State U., Raleigh, NC 27695
<http://www.cbl.ncsu.edu/>

Abstract – *Traditionally, the algorithms that optimize placement of cells in a VLSI layout rely on a cost function that includes a distance metric. The choice of a cost function may limit the choices in devising new, more efficient and more effective algorithms. The metric we propose relies on the interval graph of net nodes, induced by each specific placement of cells into a row. Nominally, the metric associated with this graph is the chromatic number (χ). The new cost function, proposed in this paper, is the hypercrossing number (HC), defined by the cardinality of the edge-set of the interval graph. This metric relates to the edge crossing number (EC) in the corresponding bipartite graph, depends on cell and net order only, and is independent of the target technology.*

Currently, there is no explicit algorithm to place cells such that the hypercrossing number is minimal, but edge crossing minimization algorithms are available. We demonstrate, experimentally and with statistical significance, the following results for the linear placement problem: (1) cost functions based on distance metric and technology-specific wire length measured after routing have a positive linear correlation with the hypercrossing number (with a coefficient of correlation nearly 100%); (2) total wiring area, measured after routing, is highly correlated with the product of the hypercrossing and chromatic number; (3) the edge crossing number in the corresponding bipartite graph model is highly correlated with the hypercrossing number; (4) relatively simple heuristic algorithms that minimize the edge crossing number in bipartite graphs also effectively minimize the hypercrossing number, producing linear placements of quality that can be better than or comparable to state-of-the-art placement algorithms – but at a much lower cost of computation.

Keywords: cell placement optimization, crossing number and wire length minimization.

1 Introduction

In submicron-technology, the interconnect dominates the performance and the area of VLSI circuits and systems on the chip. Traditionally, the algorithms that optimize placement of cells in a VLSI layout rely on a cost function that includes a distance metric. Algorithms that use distance metric as a cost function include approaches as diverse as simulated annealing [1, 2, 3], quadrisection [4], and recursive and quadratic programming [5, 6, 7, 8, 9, 10, 11]. Comparisons between different wire length objectives are presented in [12, 13]. Models and algorithms for accurately estimating the average interconnect length for both random and optimized placements have been presented in [14]. Abstract placement models, such as presented in [15], also aim to minimize the channel density.

Algorithms based on edge crossing number in graphs are few in the VLSI field. The edge crossing number and wiring area have been investigated to find lower bounds on the layout area and the maximum edge length of a variety of computational networks [16]. In [17], authors attempt to minimize the total

* F. Brglez and D. Ghosh have been supported by contracts from the Semiconductor Research Corporation (94-DJ-553), SEMATECH (94-DJ-800), DARPA/ARO (P-3316-EL/DAAH04-94-G-2080), and (DAAG55-97-1-0345), and a grant from Semiconductor Research Corporation.

“Permission to make digital/hard copy of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage, the copyright notice, the title of the publication and its date appear, and notice is given that copying is by permission of CBL. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.”

number of crossings by eliminating the redundant ones. Given a global routing, the problem of ‘properly’ distributing the set of edge crossings among the regions has been studied in [18]. Edge crossing minimization as a post-optimization problem after global routing is reported in [19].

However, in fields as diverse as economics, social sciences, mathematics and computer science, research on edge crossing number minimization in graphs has been carried out for many years. Edge crossing theory has been developed to improve the readability of hierarchical structures [20]. The problem of placing vertices for minimum edge crossing number is NP-complete, even for 2-layer graphs [21]. Graph drawing packages can now be obtained from the Web [22], [23]. A textbook, bringing this research up-to-date, is now available [24].

The choice of a cost function may limit the choices in devising new, more efficient and more effective algorithms. The metric we propose relies on the interval graph of net nodes, induced by each specific placement of cells into a row. Nominally, the metric associated with this graph is the *chromatic number* (χ). The new cost function, proposed in this paper, is the *hypercrossing number* (HC), defined by the cardinality of the edge-set of the interval graph. This metric relates to the *edge crossing number* (EC) in the corresponding bipartite graph, depends on cell *and* net order only, and is independent of the target technology.

Currently, there is no explicit algorithm to place cells such that the hypercrossing number is minimal. However, we demonstrate, experimentally and with statistical significance, the following results for the linear placement problem:

1. cost functions based on distance metric and technology-specific wire length measured after routing have a positive linear correlation with the hypercrossing number (with a coefficient of correlation nearly 100%);
2. total wiring area, measured after routing, is highly correlated with the *product* of the hypercrossing and chromatic number;
3. the edge crossing number in the corresponding bipartite graph model is highly correlated with the hypercrossing number;
4. relatively simple heuristic algorithms that minimize the edge crossing number in bipartite graphs also effectively minimize the hypercrossing number, producing linear placements of quality that can be better than or comparable to state-of-the-art placement algorithms – but at a much lower cost of computation.

Specifically, our experiments are conducted on a number of equivalence class circuits, with 100 circuits in each class, including the isomorphism classes and the mutant classes of increasing size [25]. We rely on two representative placement algorithms using a cost function that includes a distance metric: one based on quadrisection for standard cell layouts [4, 26], and one based on a state-of-the-art simulated annealing for FPGA layouts [3, 27]. While heuristics to minimize hypercrossing number directly are yet to be invented, we rely on two representative placement algorithms that minimize the edge crossing number in graphs only: the *median heuristic* [24], and the *adaptive insertion++ heuristic* [28]. We record results of placement by each algorithms in four ways. First, we record the technology-independent parameters such as edge crossing number EC , the hypercrossing number HC , and the chromatic number χ . Second, we route the given placement with a standard cell router in OASIS [26] and record the total wire length, number of tracks, and the total *wiring area*. Third, we route the given placement with a FPGA router in VPR [3, 27] and record the total wire length, number of tracks, and the total *wiring area*. Fourth, we record CPU time to execute *placement only* algorithm for each circuit instance. All algorithms are implemented and executed serially on the same CPU (SUN Ultra 2). A number of statistical charts summarize the result of each experiment, including correlations of interest between the reported parameters.

The paper is organized into following sections: (2) Background and Motivation; (3) Design of Experiments; (4) Experimental Results; (5) Conclusions.

2 Background and Motivation

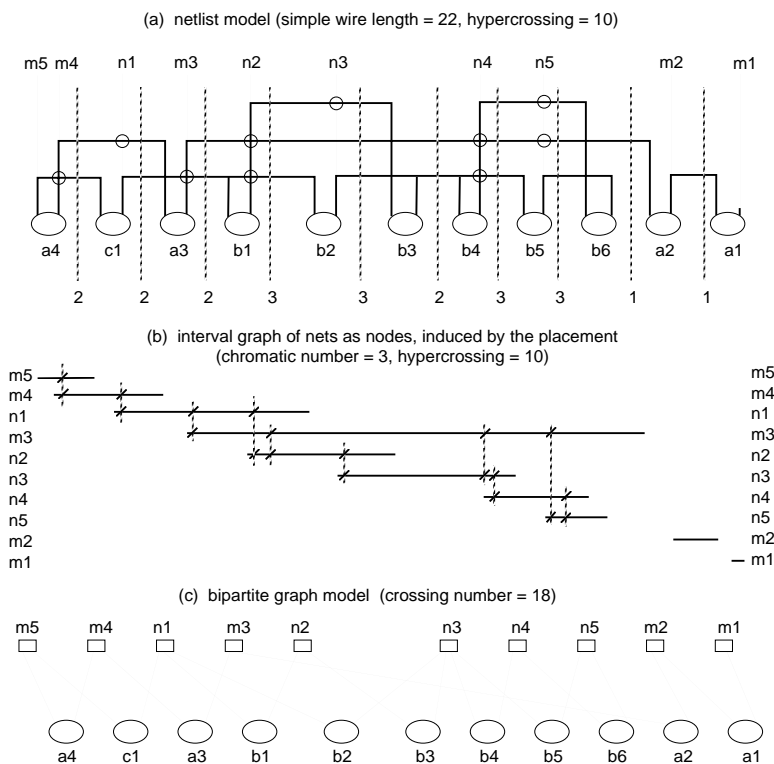


Figure 1: Graph models and key parameters associated with a linear cell placement.

Consider a simple netlist with 11 cells and 10 nets. Figure 1 illustrates three graph models related to the linear placement of such a netlist:

- (a) traditional netlist model, here with nets labeled (m_i, n_i) and routed rectilinearly in three tracks for visual simplicity above the cells, labeled as (a_i, b_i, c_i) . Without loss of generality, arc-based grid-less routing could be used to illustrate the same points;
- (b) interval graph of nets as nodes, induced by the specific placement (net nodes are represented with solid lines, edges are shown as dotted lines);
- (c) a 2-layer bipartite graph model of net nodes and cell nodes. A specific placement induces the order of both the net nodes and cell nodes (net nodes are represented as squares, cell nodes are ovals, and edges are represented as straight lines).

Five parameters can be readily associated with these linear placement models:

SWL (simple wire length): Simple wire length is a measure of total interconnection length based on a weighted sum of all net cuts. A net cut corresponds to the number of (weighted) nets cut by the plane separating two adjacent cells. Assuming that all cells are placed on a unit grid, and that all nets have unit weight, the simple wire length for the example in Figure 1(a) is then 22. This measure was used to report results of optimized linear placements in [29, 30].

HC (hypercrossing number): Formally, hypercrossing number HC is defined by the cardinality of the edge-set of the interval graph of nets, induced by a specific linear placement. For the example in Figure 1(b), the hypercrossing number is 10. To illustrate a physical interpretation of hypercrossing, we extend each net in Figure 1(a) with an additional *anchor pin*, and connect the pin with a vertical

net segment to the horizontal net segment. We place this pin at a height that exceeds the topmost horizontal segment. This extension of each net has no effect on the simple wire length (SWL) as defined earlier. However, the extension does induce a unique set of net crossings (hypercrossings), shown as 10 circles in Figure 1(a), that have a one-to-one correspondence with the edge-set of the interval graph of nets in Figure 1(b). Notably, the cardinality of the edge-set is invariant to the net order.

χ (chromatic number): Chromatic number χ is the traditional parameter of interest for an interval graph and can be found in linear time with a *left-edge* algorithm [31]. For an interval graph induced by a linear placement, the chromatic number gives an upper bound on the number of routing tracks that consist of non-overlapping routing segments. For the example in Figure 1(b), the chromatic number is 3.

EC (edge crossing number): We associate an edge crossing number EC with a bipartite two-layer graph that is derived from a netlist model in Figure 1(a) as follows: (1) sort all nets in the interval graph (induced by linear placement) by their left-edge coordinate, i.e. the minimum value of the interval; (2) replace each net with a net node and place the resulting nodes onto the line $y = 1$ in the order determined by the sort of the interval graph; (3) place all cell nodes onto the line $y = 0$ in the order induced by linear placement; (4) connect net nodes and cell nodes with straight line edges in accordance with the adjacency list.

The edge crossing number EC is the total number of line intersections induced by the given placement of net nodes *and* cell nodes. For the example in Figure 1(c), the edge crossing number is 18.

$HC \times \chi$ (area index): Area index is a composite based on two parameters associated with the interval graph: hypercrossing number HC and χ (chromatic number). While the hypercrossing number alone correlates extremely well with the total wire length reported by a given router, our experiments will also demonstrate that the product of hypercrossing and chromatic number also significantly improves the linear correlation of area measured after placement and routing, compared to correlating the measured area with hypercrossing number alone.

Illustrative Experiments. The simple circuit in Figure 1 is deceptively hard for a number of state-of-the-art placement algorithms. Intuitively, one expects that a better linear placement can be found by placing the 11 cells in different order, minimizing the simple wire length, hypercrossing, chromatic number, or the edge crossing number. Questions that arise include: what is the “best placement” in terms of these parameters and, would minimizing a specific single parameter minimize them all?

We illustrate the variability of performance of several algorithms by designing a series of experiments with the *isomorphism equivalence class* of 100 netlists based on the simple netlist in Figure 1. This approach is fundamentally different from the traditional approach where a number of algorithms is tested on a single instance of a circuit representation. Intentionally, we devise, for any given netlist, an isomorphism class as follows:

P1: create a new instance of the netlist by randomizing the order of all nodes in the netlist;

P2: assign random names to all nodes in each instance of the netlist.

Properties **P1** and **P2** are essential to good experimental design and must be maintained universally for all equivalence classes, not only the isomorphism class. The purpose of **P1** is clear. Without **P2**, some programs that rely on hashing the input data may *unknowingly undo* the randomization of input presentations and confound the experiments. Important lessons on this subject have been learned and reported in [25, 32]. We introduce other other equivalence classes, also used in our experiments, later in the paper.

We performed a series of five experiments on the isomorphism equivalence class of circuit in Figure 1. These experiments involve distinct algorithms and we refer to each algorithm as a treatment. Treatment numbers 0–19 come from earlier experiments on edge crossing minimization as reported in [28]. The treatments we use here are:

- Designated as TR0, this experiment is equivalent to a random placement since we are placing the node in the order given by the netlist;

- Designated as TR7, this placement minimizes edge crossings, based on the *median* heuristic [24];
- Designated as TR19, this placement minimizes edge crossings, based on the *adaptive insertion++* heuristic [28];
- Designated as TR20, this placement minimizes a distance-based metric, based on the standard-cell *quadrisection* heuristic [4, 26];
- Designated as TR21, this placement minimizes a distance-based metric, based on the FPGA *simulated annealing* heuristic [3, 27].

| | TR 0 | | TR 7 | | TR 19 | | TR 20 | | TR 21 | |
|------------------|-------|------|------|------|-------|------|-------|------|-------|------|
| | mean | s.d. | mean | s.d. | mean | s.d. | mean | s.d. | mean | s.d. |
| SWL | 42.4 | 10.6 | 15.3 | 2.13 | 14 | 0 | 16.9 | 2.24 | 15.2 | 2.07 |
| χ | 6.6 | 0.99 | 2.32 | 0.47 | 2 | 0 | 2.4 | 0.49 | 2.26 | 0.44 |
| HC | 32.2 | 5.44 | 5.33 | 1.77 | 4 | 0 | 8.13 | 2.65 | 5.72 | 1.98 |
| EC | 60.1 | 10.7 | 6.84 | 6.85 | 4 | 0 | 12.3 | 5.89 | 10.1 | 6.79 |
| $HC \times \chi$ | 216.8 | 65.9 | 13.1 | 7.0 | 8 | 0 | 20.9 | 9.02 | 13.5 | 7.18 |

(a)

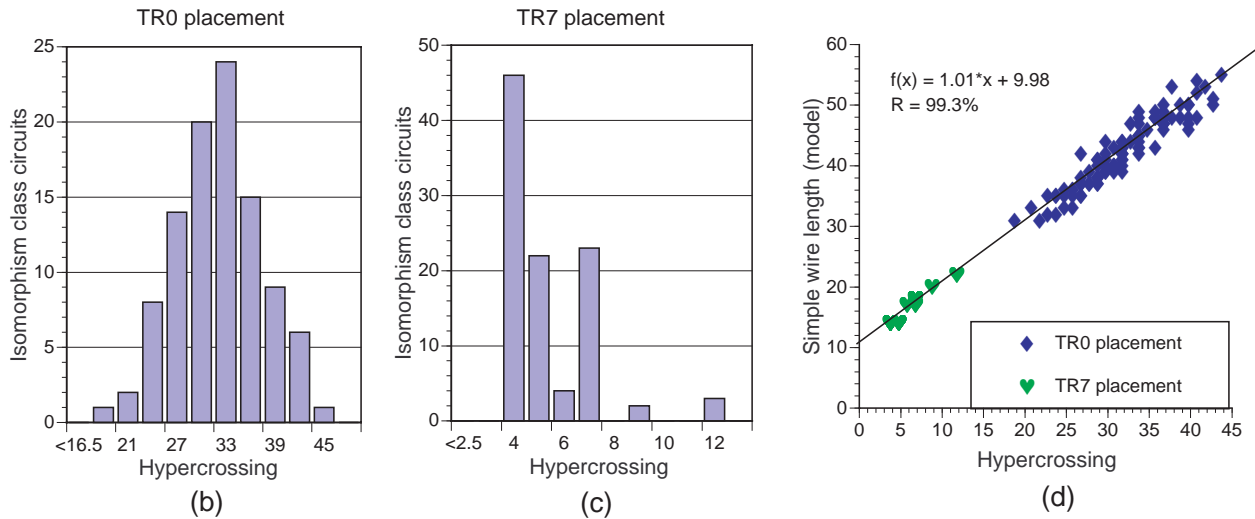


Figure 2: Summary of experiments with the isomorphism class of the circuit example in Figure 1.

The results of this series of experiments are summarized in Figure 2. All graph parameters defined in Figure 1 are tabulated for each of the experiments. The mean and standard deviation are based on instances of 100 circuits in the isomorphism equivalence class. A number of observations one can infer from these results may be surprising:

- The significant variance of reported means for TR0, an effectively random placement, is expected, and as shown in Figure 2(b), the distribution of hypercrossing is nearly normal.
- Algorithms corresponding to TR7, TR20, TR21 show non-trivial variance of reported means – for all parameters listed. The span of extreme variations can be significant. For example, the minimum and maximum simple wire length (SWL) as reported by TR21 (simulated annealing) varies from 14 to 22. Initiating a single experiment with a ‘good’ starting point can ‘reduce’ the cost function by some 32% without even ‘improving’ the algorithm. This span increases as the size of circuits increases, as shown in later experiments.

- Only the algorithm corresponding to TR19 exhibits the ‘expected ideal’ behavior when tested with an isomorphism class: variance of 0 for each reported mean! Furthermore, the placement reported is optimal in terms of the edge crossing number of 4 [28].
- As shown in Figure 2(c), the distribution of hypercrossing corresponding to TR7 does reach the minimal value 46 times out of 100. Similar distributions occur with the other treatments (except for the ideal TR19 and the significantly worse TR20) and with other cost functions.
- For this simple example, the reported simple wire length of 14, chromatic number of 2, hypercrossing of 4 and area index of 8 are also optimal.
- The near-perfect correlation between the hypercrossing and simple wire length as reported for both TR0 (random placement) *and* TR7 (median heuristic) is revealing – and is clearly independent of the placement algorithm used. The slope of the line is the same for several different circuit classes of as well. The relationship suggested by the data is $SWL = (1 + \epsilon)HC + n - 1$, where n is the number of cells. It appears that the circuits we’ve considered tend have one net continuing past each cell in addition to the two connecting it as we move from left to right. The small ϵ factor suggests that with more hypercrossings, there may occasionally be more than one continuing net.
- The relationship between hypercrossings and edge crossings is more complex. It makes sense that $EC = \alpha \cdot HC$, but α depends on two main factors: (i) the average number of cells per net (α will be slightly larger than this average when the EC solution quality is low to moderate; if this average is significantly greater than 2, there should be a quadratic dependence on it, but this does not show up in our sample circuits), and (ii) the quality of the EC solution: α will decrease significantly as EC approaches optimum; the linear correlations in our data the tables account for this decrease in α with a negative added term that is related to the size of the graph.

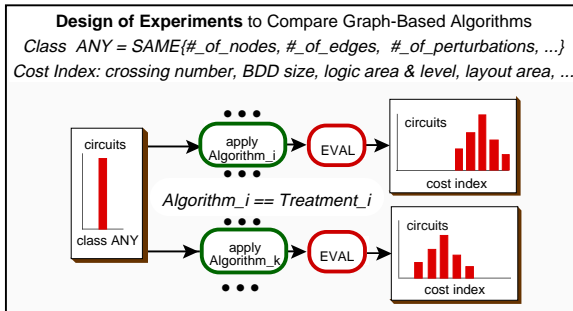
Our experiments strongly suggest that it is in some sense easier to focus on the edge crossing metric when optimizing placement for any or all of the other metrics. At any intermediate stage of a heuristic that iteratively improves placements based on bipartite edge crossing, the net nodes serve as anchors to keep each net tied together as it moves to separate itself from other nets. The strong correlation between edge crossing and metrics that impact the final routing makes it possible to use simpler, better, and/or less computationally intensive edge crossing minimization algorithms as a substitute for existing placement algorithms in state-of-the-art physical design tools. In what follows we report experimental results that are a first step in that direction.

3 Design of Experiments

Two of the fundamental principles of experimental design are *randomization* and *replication*. We adopt these principles for the experimental evaluation of heuristics by (1) creating an *equivalence netlist class*, and (2) repeating the experiments for each member in the class.

Using a simple example in the preceding section, we have already demonstrated merits of replicating the experiments for circuits in the isomorphism equivalence class, making a strong case for formalizing the *Design of Experiments* (DoE) and thereby providing the scientific basis for the performance evaluation of algorithms in CAD. There is no merit in evaluating and comparing performance of two algorithms on the basis of a single instance of a reference circuit. Making comparisons across a few unrelated reference circuits has no statistical significance either. Variations on the order of 50% are not unusual and have been observed and reported for a number of cases that only involve circuits in the isomorphism equivalence class [25, 32, 33, 34]. Study of equivalence classes other than isomorphism is required to gain additional insights and we conclude this section with a brief introduction to such classes. For the context of this paper, the basic abstractions for DoE include [33]:

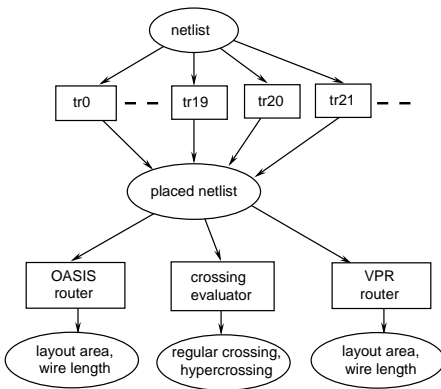
1. an equivalence class of experimental subjects, eligible for a treatment;
2. application of a specific treatment;
3. statistical evaluation of treatment effectiveness.



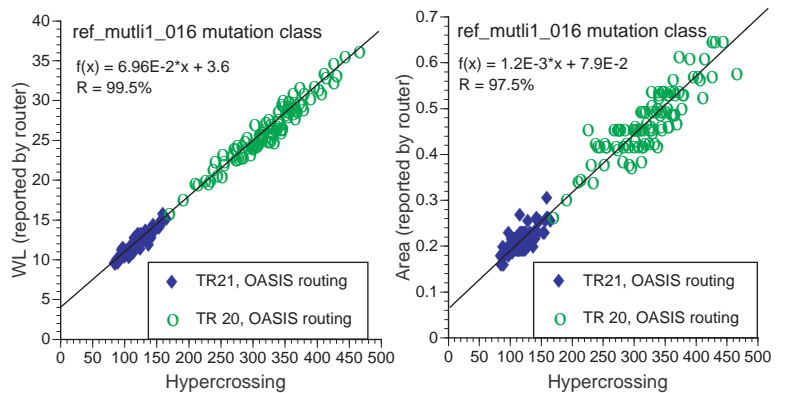
| Legend | |
|-----------------------------|--|
| <i>natural</i> | node order given by the input graph |
| <i>BFS</i> | breadth-first order |
| <i>GBFS</i> | 2-pass guided breadth-first order |
| <i>median</i> | as described in [24] |
| <i>barycenter</i> | as described in [24] |
| <i>median/barycenter</i> | combines median and barycenter |
| <i>adaptive insertion</i> | inserts each vertex optimally |
| <i>adaptive insertion++</i> | adaptive insertion alternated with median/barycenter |
| <i>OASIS</i> | OASIS standard cell placement and routing as described in [26] |
| <i>VPR</i> | FPGA placement and routing as described in [3] |

| Treatment | Initial Placement | Final Placement |
|-----------|-------------------|----------------------|
| 0 | natural | none |
| 7 | BFS | median |
| 19 | GBFS | adaptive insertion++ |
| 20 | OASIS | OASIS |
| 21 | VPR | VPR |

Figure 3: Design of experiments to compare the placement produced by different algorithms (treatments).



(a) Workflow for treatments



(b) Hypercrossing and wire length correlation

Figure 4: Workflow for treatments and typical correlations.

Here, a *treatment* is synonymous with a *heuristic* and an equivalence class of experimental subjects is synonymous with a netlist equivalence class. Figure 3 illustrates these abstractions in a generic flow. In the context of this paper, the cost indexes refer to the cost function minimized by any given placement algorithm: either a model of total wire length or a model of total edge crossing. The companion table shows a list of 5 placements (treatments) for which we have designed experiments reported in the preceding section and in the rest of this paper.

For each treatment, we measure and report the parameters of interest illustrated in Figure 1 and defined in the preceding section. The simplified flow of the complete experimental design is shown in Figure 4(a). For each netlist, in each equivalence class, we execute:

- each of treatments 0, 7, 19, 20, 21 to place the netlist;
- invoke technology-specific channel routers OASIS and VPR to generate the layout (for standard cell layout and FPGA respectively) and report the *total wire length* and *routing area*;
- invoke a technology-independent `crossing_evaluator` to report the *edge crossing*, *hypercrossing* and *simple wire length model* in the placement.

The example in Figure 4(b), based on data for 100 instances of the mutation class of `ref16_multi1_16`, demonstrates the correlations between the wire length and hypercrossing distributions, and the wiring area and the hypercrossing distributions for two treatments, Treatment 20 and Treatment 21 – both evaluated by a common standard cell placement router in OASIS. While both treatments are very different, the respective correlation coefficients are very high: 99.5% and 97.5% respectively – and note that the Treatment 21, designed to place FPGA cells, is actually much better than Treatment 20, designed to place standard cells. In addition, the coefficient of correlation is clearly independent of the treatment we apply. The questions addressed in the remainder of this paper include: (1) what correlations can we expect with equivalence classes other than this class, and (2) how will correlation scale as we vary the topology and increase the size of the reference circuits for each class.

Mutation Equivalence Classes. The concept of the isomorphism class, introduced informally in the previous section, demonstrated merits and principles for the design of experiments. Here, we formalize the isomorphism class such that it leads naturally to the more generalized concept of the *mutation class* for bipartite graphs.

Let $G_r = (V_0, V_1, E)$ designate the *reference bipartite graph*. A *characteristic signature* can be any number of mappings based on parameters that relate to the size and distribution of vertices and edges in G_r . For example, to support the definition of a mutation class later in this section, we introduce a signature σ_{mut} ,

$$\sigma_{\text{mut}}(G_r) = \{|V_0|, (\text{dist}|V_1|), (\text{dist}(E))\} \quad (1)$$

where $(\text{dist}|V_1|)$ denotes the distribution of vertices in V_1 , classified in terms of edges incident at each vertex, and $(\text{dist}(E))$ denotes the distribution of edges in E classified in terms of *connected components* that constitute G_r .

We create a *perturbation graph* G'_r from G_r as follows: (1) we take a copy of G_r and randomly select and remove a fraction q of edges from E and put them into an empty urn, (2) we pick edges from the urn and randomly connect vertices V_0 and V_1 , until the urn is empty.

We consider each placement of $G = (V_0, V_1, E)$ as a *presentation* $\langle G, \pi_0, \pi_1 \rangle$, where π_i is a permutation of V_i . Two equivalence classes we use in our experiments are the isomorphism class \mathcal{G}_{iso} and the mutation class \mathcal{G}_{mut} defined as

$$\mathcal{G}_{\text{iso}} = \{\text{presentation of } \langle G_r, \pi_0, \pi_1 \rangle\} \quad (2)$$

$$\mathcal{G}_{\text{mut}} = \{\text{presentation of } \langle G'_r, \pi_0, \pi_1 \rangle, |\sigma_{\text{mut}}(G_r)\} \quad (3)$$

where π_i are strictly *random permutations*.

Remarks. A presentation of the perturbation graph, $\langle G'_r, \pi_0, \pi_1 \rangle$, is not necessarily in the mutation class. As expressed in (3), in order to belong to the mutation class, each perturbation graph must also satisfy the signature σ_{mut} in (1) – a non-trivial procedure which has been generalized for arbitrary netlists and is described in [25]. Both the isomorphism class \mathcal{G}_{iso} and the mutation class \mathcal{G}_{mut} satisfy the signature σ_{mut} in

(1): both have the same number of nodes, edges, and connected components – as well as the *distribution* of cell types, e.g. all nodes in V_1 have p_i 2-pin nodes, q_i 3-pin nodes, r_i 4-pin nodes, etc. The only random variable in \mathcal{G}_{mut} (relative to \mathcal{G}_{iso}) is number of edges assigned to nodes in V_0 , i.e. the fanout of net nodes, *bounded* by the signature. Satisfying the signature also implies that no graph instance, drawn at random from the equivalence class, will ever have any isolated nodes.

4 Experimental Results

Due to space constraints, we present only a few representative results in this section. The experiments are based on the model shown in Figure 3(a). We use Treatments 0, 7, 19, 20 and 21 to place the circuits and then both the VPR and OASIS router to route them. Results presented here compare the performance of the placement algorithms and determine the correlations between the various cost functions and the layout parameters such as wire length and wiring area.

The test cases comprise of isomorphism and mutation classes of circuits of varying complexity. These are representative of connected components extracted from netlists available in [35]. The netlists can be broadly classified into two groups:

- circuit graphs with a single connected components – denoted by `ref_multi1_n`, $n=2,4,8,16,32$, with $4n + 3$ cells and $4n + 2$ nets. The equivalence class graphs thus have 32, 56, 104, 200, 392 pins respectively.
- circuit graphs with multiple connected components – denoted by `ref_multi1_008Xm`, $m=1,2,3,4$, which is a concatenation of 1, 2, 3, 4 circuits of type `ref_multi1_008`. The equivalence class graphs thus have 104, 208, 312 and 416 pins, respectively.

We generate isomorphism and mutation classes of each reference graph (`ref_multi1_n` or `ref_multi1_008Xm`) as described in the preceding section. For each netlist in each equivalence class we execute the workflow as shown in Figure 3(a), which serially executes each of treatments 0, 7, 19, 20, 21 on each netlist and tabulates results for all parameters of interest. Our results are summarized in statistical charts as shown in Figures 5 to 11.

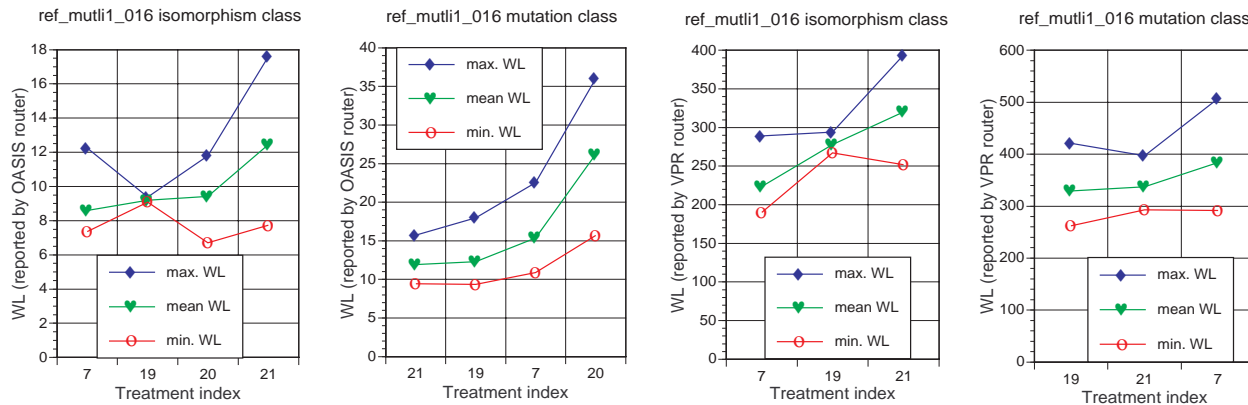


Figure 5: Wire length, as reported by two routers (OASIS and VPR), for two equivalence classes and four placements (Treatments 7, 19, 20, 21).

Figure 5. The experiments summarized in Figure 5 are designed to reveal:

- variability and performance of the five placement algorithms (Treatments 0,7,19,20,21);
- the sensitivity of each treatment relative to the two input data classes: the isomorphism class and the mutant class;

- the relative ranking of all treatments with respect to total wire length for both equivalence classes.

Data, as arranged in Figure 5 (sorted on the average total wire length for each treatment), lead to following observations and tentative conclusions:

- Treatment 0, representing the case of *no treatment* or a *placebo treatment*, is always far worse than other treatments. Treatment 0 results are not plotted to permit rendering other results on an expanded scale.
- For the isomorphism class and routing by OASIS router, Treatment 19 results in the same wire length for each circuit which should be the case for isomorphic circuits. However, all other treatments show a wide variation in performance with Treatment 21 being the worst.
- For the mutation class and routing by OASIS router, we see a variation over the class for all the treatments which is expected since all the circuits are different. Treatment 20 is the worst performer here. Performances of Treatment 19 and Treatment 21 are comparable (statistically indistinguishable), although Treatment 21 has a slightly lower mean than Treatment 19.
- For routing by the FPGA router VPR, same observations are repeated for the isomorphism class. Treatment 19 shows same performance for all the circuits while all other treatments have variations. Treatment 20 data are not included here since we found it to be consistently performing worse than Treatment 19.
- For the mutation class and routing by VPR, Treatment 19 is marginally better than Treatment 21, however, the differences are not significant.

The trend seen in Figure 5 are mostly repeated for other equivalence classes too. Treatment 19 shows the least variation for the isomorphism classes and has the best overall results although Treatment 21 does not lag far behind. Treatment 21 is sometimes marginally better than Treatment 19 although the differences are not statistically significant.

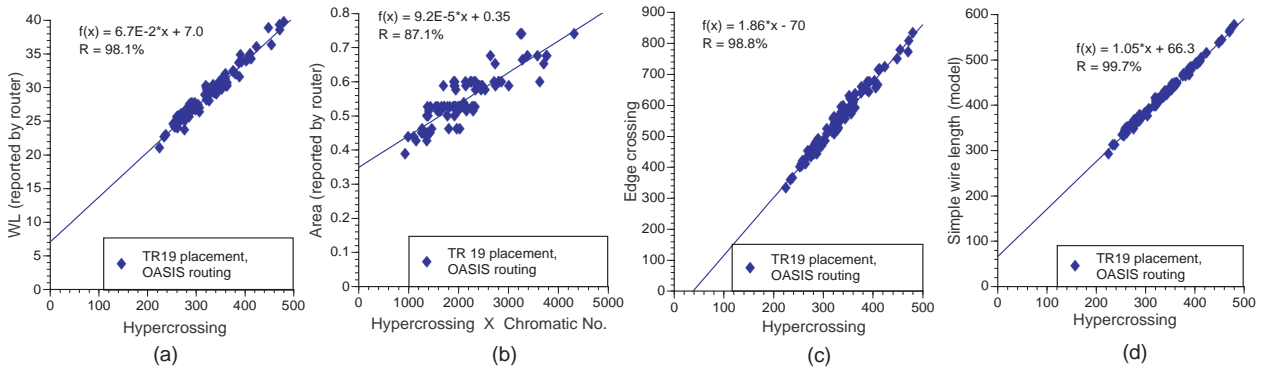


Figure 6: Experiments with the `ref_multi1_032` mutation class and Treatment 19 after routing with OASIS: a correlation report.

Figure 6. The experiments summarized in Figure 6 are similar to the correlation experiment introduced in Figure 4(b). The basic difference is that instead of Treatment 21 being applied to the mutation class of circuit `ref_multi1_016`, it is Treatment 19 applied to the mutation class of circuit `ref_multi1_032`.

The main objectives of these experiments are:

- to evaluate the correlation, for a given placement, between the total hypercrossing in the graph model and the total wire length and area, after rectilinear routing.
- to examine the correlation between the edge crossing in the graph and the hypercrossing.
- to examine the correlation between the simple wire length model of [30] and the hypercrossing.

Data, as arranged in Figure 6, lead to following observations and tentative conclusions:

- Figure 6(a) depicts excellent correlation between the hypercrossing and the wire length reported by the router. It is clear that hypercrossing can be used to predict the wire length with good accuracy. Notably, the hypercrossing is a topological property of the placement and it is independent of the technology in which the placement is going to be routed. This topological parameter can be used as a cost function that can be optimized with a view to optimizing the layout wire length, irrespective of the target technology. Although not evident from this figure, the correlation between wire length and hypercrossing is found to hold good irrespective of the treatment used for placement or the router used for routing.
- In Figure 6(b), we see very good correlation between the routing area (as reported by the OASIS router) and the product of the hypercrossing and the chromatic number of the net interval graph. Area, being a two-dimensional parameter, correlates better with this product while the wire length correlates well with the hypercrossing.
- In Figure 6(c), we see excellent correlation between the hypercrossing and the edge crossing (in the bigraph). It implies that one can minimize hypercrossing indirectly by minimizing the regular wire crossing. This observation is important since we do not have an algorithm to minimize hypercrossing directly. On the other hand there are many good heuristics for edge crossing minimization, Treatment 19 being one of the best known to date [28]. Thus minimizing edge crossing is a good heuristic for hypercrossing minimization, and in turn, the layout wire length.
- Finally, in Figure 6(d), we see excellent correlation between the hypercrossing and the simple wire length modeled in [30] as a cost function for linear placement. The near perfect correlation implies that we can substitute hypercrossing for the traditional cost function of simple wire length.

Figure 7. The experiments summarized in Figure 7 are similar to the correlation experiment introduced in Figure 6. The basic difference is that instead of Treatment 19 being applied to the mutation class of circuit `ref_multi1_032`, it is Treatment 21 that is applied to the same mutation class. Regardless of the fundamental change of the placement algorithm applied to generate this data, we observe:

- The correlation charts in Figure 6(a) and Figure 7(c) show comparable (high) correlation of hypercrossing to wire length reported by OASIS router.
- The correlation charts in Figure 6(c) and Figure 7(f) show comparable (high) correlation of edge crossing to hypercrossing reported by the `cross_evaluator`.
- The correlation charts in Figure 6(d) and Figure 7(g) show comparable (high) correlation of simple wire length to hypercrossing length reported by the `cross_evaluator`.
- The correlation charts in Figure 6(b) and Figure 7(d) show comparable (high) correlation of area index (product of hypercrossing and chromatic number) to wiring area reported by OASIS router.
- Correlations specific to Treatment 21 include:
 - Figure 7(a), with correlation coefficient of 95.9% of *cost model* optimized directly by Treatment 21 relative to wire length reported by OASIS router – whereas the correlation coefficient for the hypercrossing (for the same placement) relative to wire length is reported at 96.6% in Figure 7(c).
 - Figure 7(e), with correlation coefficient of 98.8% of hypercrossing relative to the *cost model* optimized directly by Treatment 21.
 - Figure 7(b), with correlation coefficient of 78.3% of *cost model* optimized directly by Treatment 21 relative to the wiring area reported by OASIS router – whereas the correlation coefficient for the area index (product of hypercrossing and chromatic number, for the same placement) relative to wire length is reported at 85.3% in Figure 7(d). This demonstrates that both hypercrossing *and* chromatic number are important when correlating to the wiring area after routing.

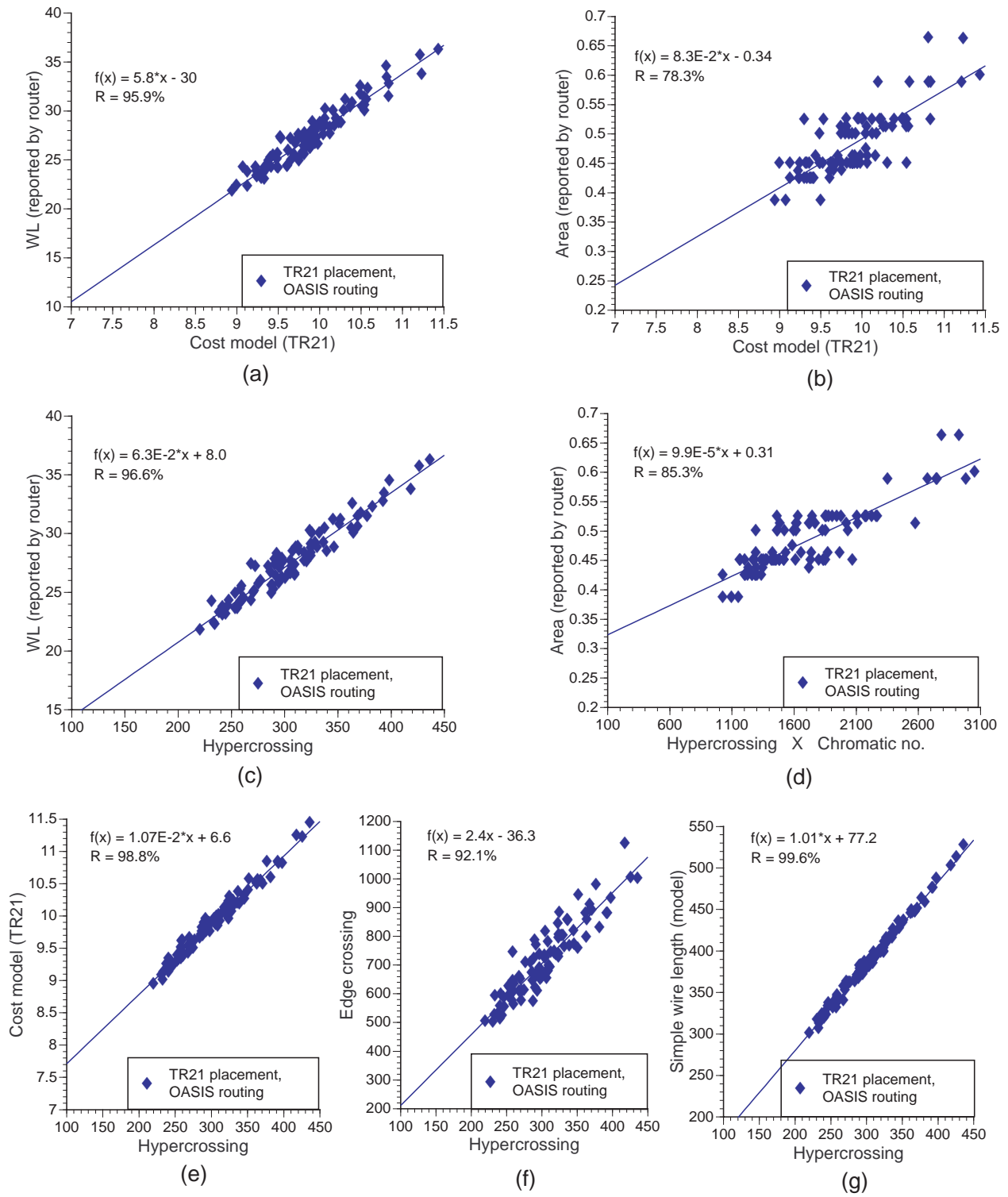


Figure 7: Experiments with the `ref_multi1_032` mutation class and Treatment 21 after routing with OASIS: a correlation report.

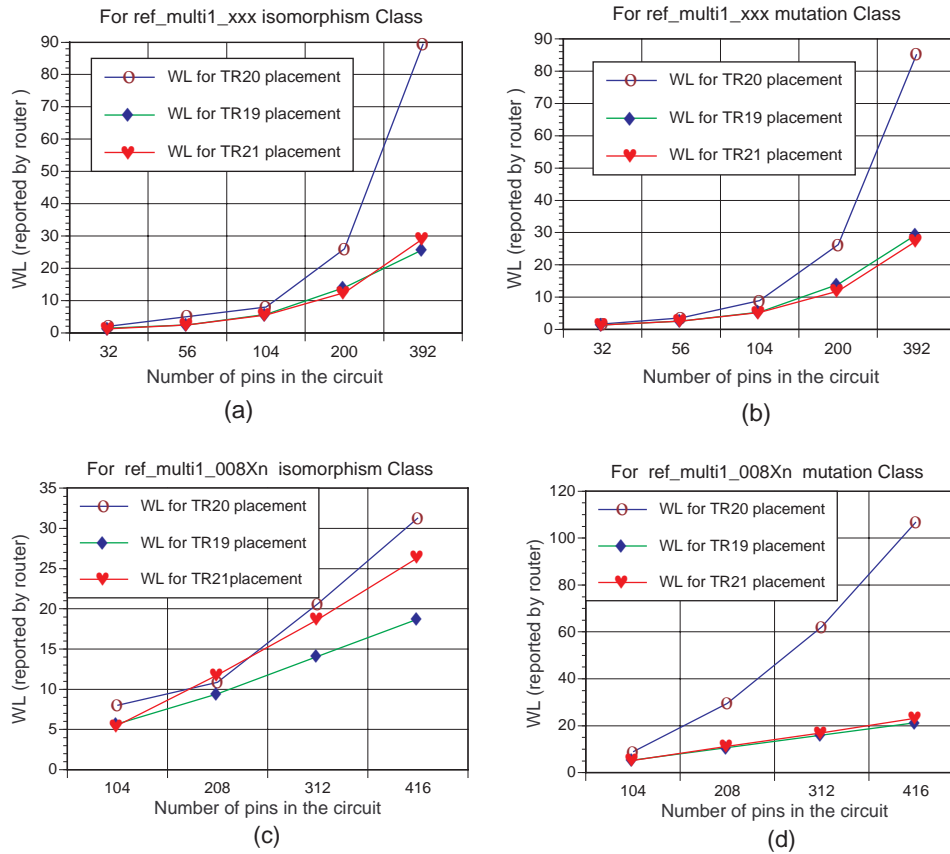


Figure 8: Experiments on asymptotic behavior with eighteen equivalence classes and three placements (Treatments 19, 20 and 21): reporting wire length after OASIS routing.

Figure 8. The experiments in Figure 8 summarize the asymptotic behavior with 18 equivalence classes and three placements (Treatments 19, 20, and 21) – all reporting wire length mean values after OASIS routing.

- Figure 8(a) shows wire length mean values for five isomorphism classes of type `ref_multi1_xxx`. While differences in Treatment 19 and 21 are minor, Treatment 20 is clearly not scaling well with the increasing circuit size in the given equivalence class.
- Figure 8(b) shows wire length mean values for five mutation classes of type `ref_multi1_xxx`. While differences in Treatment 19 and 21 are minor, Treatment 20 is clearly not scaling well with the increasing circuit size in the given equivalence class. In fact this class appears to exhibit behavior, for all algorithms, that is similar to the behavior for the isomorphism class. However this is not the case for the two classes described next.
- Figure 8(c) shows wire length mean values for four isomorphism classes of type `ref_multi1_008Xn`. Here the difference between all classes is very distinct, with Treatment 19 performing the best.
- Figure 8(d) shows wire length mean values for four mutation classes of type `ref_multi1_008Xn`. Here, the differences in Treatment 19 and 21 are minor, while Treatment 20 is clearly not scaling well with the increasing circuit size in the given equivalence class.

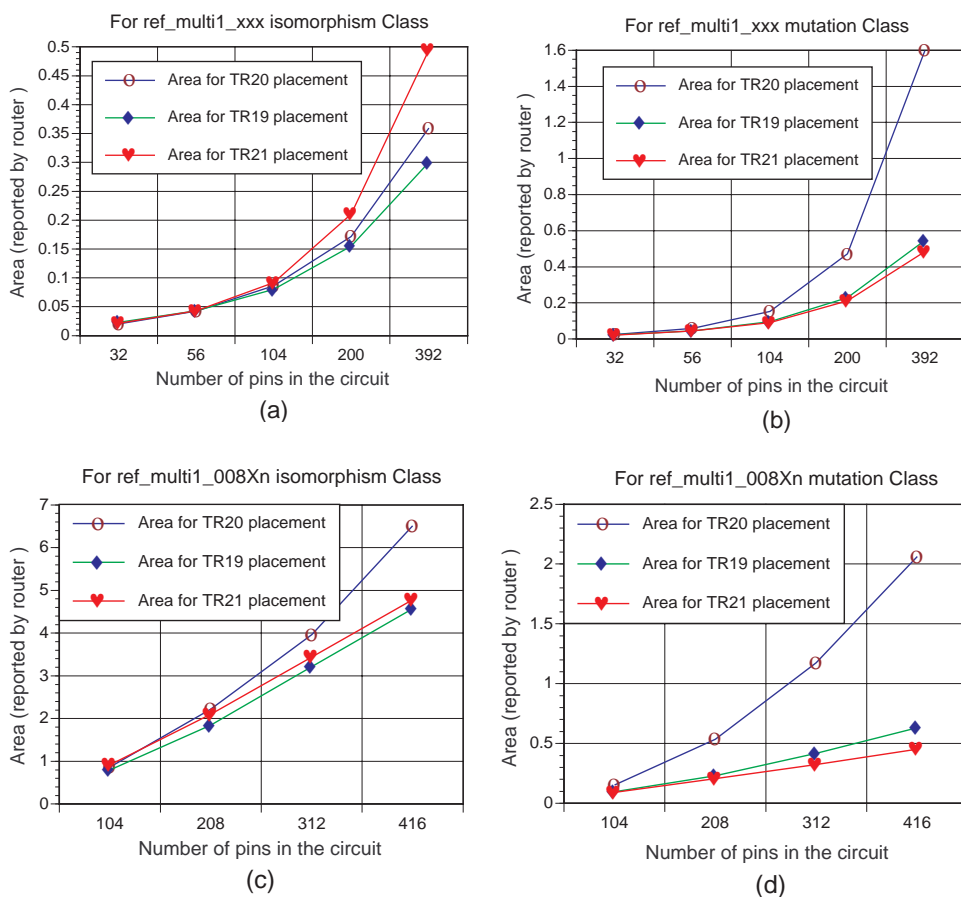


Figure 9: Experiments on asymptotic behavior with eighteen equivalence classes and three placements (Treatments 19, 20 and 21); reporting area after OASIS routing.

Figure 9. The experiments in Figure 9 summarize the asymptotic behavior with 18 equivalence classes and

three placements (Treatments 19, 20, and 21) – all reporting wiring area mean values after OASIS routing. Results reported in these charts parallel those in Figure 8 and observations and conclusions are similar.

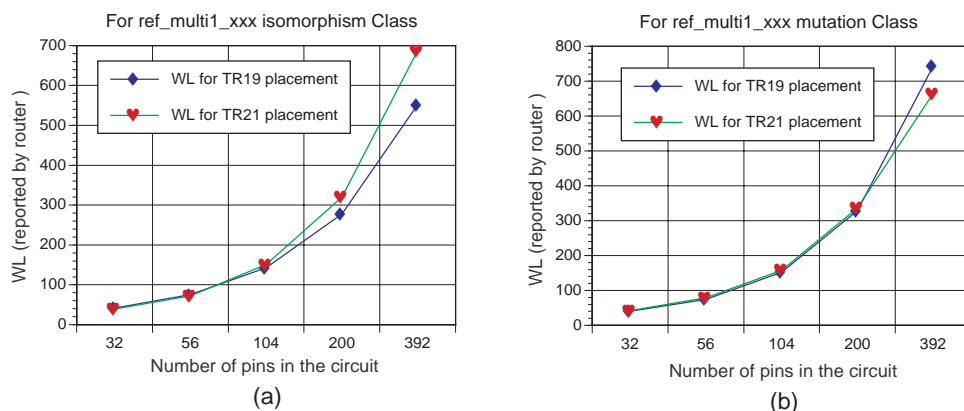


Figure 10: Experiments on asymptotic behavior with ten equivalence classes and two placements (Treatments 19 and 21): reporting wire length after VPR routing.

Figure 10. The experiments in Figure 10 summarize the asymptotic behavior with 10 equivalence classes and two placements (Treatments 19 and 21) – all reporting wire length mean values after VPR routing.

- Figure 10(a) shows wire length mean values for five isomorphism classes of type `ref_multi1_xxx`. It is quite clear that the asymptotic performance for Treatment 19 degrades less rapidly with the increasing circuit size in the given equivalence class.
- Figure 10(b) shows wire length mean values for five mutation classes of type `ref_multi1_xxx`. Here, the trend shows that the asymptotic performance for Treatment 21 may degrade less rapidly with the increasing circuit size in the given equivalence class.

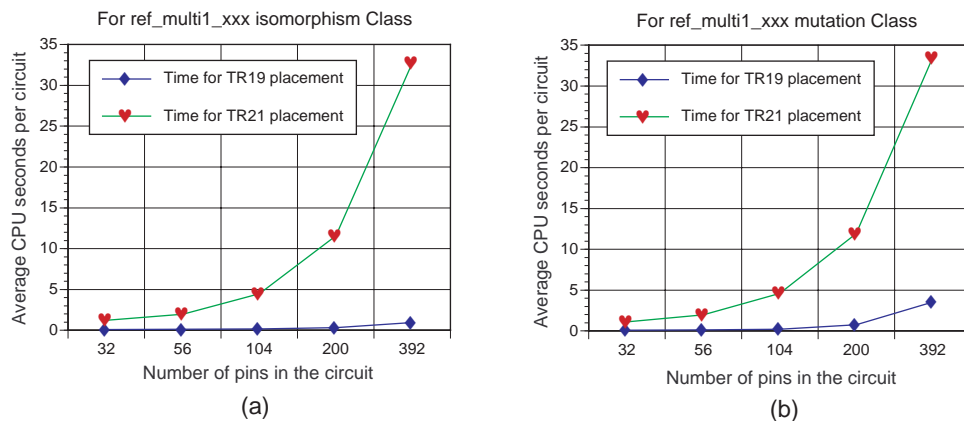


Figure 11: Experiments on asymptotic behavior with ten equivalence classes and two placements (Treatments 19 and 21): reporting average CPU time for the placement.

Figure 11. Given that Treatment 19 and Treatment 21 appear to produce placements of comparable quality, the question arises what is the CPU cost of executing each treatment. The experiments in Figure 11 summarize the asymptotic behavior with 10 equivalence classes and two placements (Treatments 19 and

21) – all reporting mean CPU time required to execute each treatment for each circuit in a given equivalence class.

- Figure 11(a) shows mean CPU time for five isomorphism classes of type `ref_multi1_xxx`. It is quite clear that the asymptotic performance for Treatment 19 is significantly better than the performance of Treatment 21, with the increasing circuit size in the given equivalence class.
- Figure 11(b) shows mean CPU time for five mutation classes of type `ref_multi1_xxx`. While for this circuit class, the asymptotic performance for Treatment 19 is not as good as the isomorphic class, its overall performance is still significantly better than the performance of Treatment 21, with the increasing circuit size in the given equivalence class.

5 Conclusion and Future Work

These experiments demonstrate that simpler and faster algorithms based on edge-crossing minimization can yield placements whose quality is good in a universal sense, independent of the target technology. Much work remains to be done, both to certify the correlations among metrics for larger classes of circuits and to develop better heuristics based on our findings. Among the open questions that remain are:

- Is there a compelling reason why heuristics that focus on the EC metric have an easier time optimizing it and the other metrics? The net nodes in the bipartite graph model serve as anchors to keep each net tied together as it moves to separate itself from other nets. A simple experiment illustrates the power of these net nodes. Using the isomorphism class of the circuit illustrated in Figure 1, we applied a simple iterative median heuristic (Treatment TR1 from [28], which is the same as TR7, but without a breadth-first search to do initial placement) to the placements generated by TR21, the simulated annealing heuristic with a distance-based metric. On small circuits there was significant improvement in EC but not in the other metrics. Limited experiments on larger circuits yielded significant improvement in all metrics. It would be interesting to see whether the simulated annealing heuristic is capable of improving placements generated by heuristics based on the EC metric.
- How easy is it to optimize the hypercrossing number directly? We suspect, but have not proven that the problem is NP-hard. Would it be easy to develop good heuristics for minimizing HC?
- Do our results carry over to denser circuits (ones with a higher ratio of nets to cells)?
- What is the exact nature of the correlations among various metrics? How do they depend on density and other parameters?
- As better heuristics for minimizing EC are developed, will they necessarily do a better job of minimizing the other metrics whenever they improve EC results?
- How can our techniques be gainfully applied to two-dimensional placement problems?

ACKNOWLEDGMENTS. We appreciate the access to the VPR tool from Prof. Jonathan Rose and the FPGA research group of University of Toronto. The tool is versatile and efficient. We also appreciate the access to the OASIS layout system. Some of the initial experiments with edge-crossing number were organized and executed by Nevin Kapur.

References

- [1] C. Sechen. *VLSI placement and global routing using simulated annealing*. Boston: Kluwer Academic, 1988.
- [2] W. Sun and C. Sechen. Efficient and Effective Placement for Very Large Circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 14(3):349–359, March 1995.

- [3] V. Betz and J. Rose. VPR: A New Packing, Placement and Routing Tool for FPGA Research. In *Proceedings of the 7th International Workshop on Field-Programmable Logic*, pages 213–222, August 1997.
- [4] P. R. Suaris and G. Kedem. A quadrisection-based combined place and route scheme for standard cells. *IEEE Transactions on Computer-Aided Design*, 8(3):234–244, 1989.
- [5] K. Doll, F. M. Johannes, and G. Sigl. Accurate net models for placement improvements by network flow methods. In *Proc. Int. Conf. on Computer-Aided Design*, pages 594–597, 1992.
- [6] J. Kleinhans, G. Sigl, F. Johannes, and K. Antreich. GORDIAN: VLSI Placement by Quadratic Programming and Slicing Optimization. *IEEE Transactions on Computer-Aided Design*, 10(3):356–365, March 1991.
- [7] G. Sigl, K. Doll, and F. Johannes. Analytical placement: A linear or quadratic objective function? In *Proc. 28th ACM/IEEE Design Automation Conference*, pages 427–432, 1991.
- [8] A. Srinivasan, K. Chaudhary, and E. S. Kuh. RITUAL: A performance driven placement algorithm for small cell IC's. In *Proc. Int. Conf. on Computer-Aided Design*, pages 48–51, 1991.
- [9] R.S. Tsay and E. Kuh. A unified approach to partitioning and placement. *IEEE Transactions on Circuits and Systems*, 38(5):521–533, 1991.
- [10] C.K. Cheung and E.S. Kuh. Module placement based on resistive network optimization. *IEEE Transactions on Computer-Aided Design*, CAD-3:218–225, 1984.
- [11] C. J. Alpert, T. F. Chan, D. J. Huang, A. B. Kahng, I. L. Markov, P. Mulet, and K. Yan. Faster minimization of linear wirelength for global placement. In *Proc. 1997 International Symposium on Physical Design*, pages 4–11, April 1997.
- [12] I.I. Mahmoud, K. Asakura, T. Nishibu, and T. Ohtsuki. Experimental appraisal of linear and quadratic objective functions effect on force directed method for analog placement. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E77-A(4):719–725, April 1994.
- [13] B. M. Reiss, K. Doll, and F. M. Johannes. Partitioning very large circuits using analytical placement techniques. In *Proc. 31st ACM/IEEE Design Automation Conference*, pages 646–651, 1994.
- [14] C. Sechen. Average Interconnection Length Estimation for Random and Optimized Placements. In *IEEE International Conference on Computer-Aided Design*, pages 190–193, November 1987.
- [15] T. N. Bui and S. Lee. On the Mincut Bipartite Arrangement Problem. In *IEEE International Conference on Computer-Aided Design*, pages 466–469, November 1987.
- [16] F. T. Leighton. New Lower Bound Techniques for VLSI. *Math. Systems Theory*, 17:47–70, 1984.
- [17] P. Groenveld. On Global Wire Ordering for Macro–Cell Routing. In *Proceedings of the 26th Design Automation Conference*, pages 155–160, 1989.
- [18] M. Marek-Sadowska and M. Sarrafzadeh. The Crossing Distribution Problem. *IEEE Transactions on Computer–Aided Design of Integrated Circuits and Systems*, 14(4):423–433, April 1995.
- [19] H.F.S. Chen and D.T. Lee. On Crossing Minimization Problem. *IEEE Transactions on Computer–Aided Design of Integrated Circuits and Systems*, 17(5):406–418, May 1998.
- [20] J. N. Warfield. Crossing Theory and Hierarchy Mapping. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC–7(7):505–523, July 1977.
- [21] M. R. Garey and D. S. Johnson. Crossing Number is NP-complete. *SIAM J. Algebraic Discrete Methods*, 4:312–316, 1983.

- [22] E.R. Gansner, E. Koutsifios, S.C. North and K.P. Vo. A Technique for Drawing Directed Graphs. *IEEE Trans. Software Engg.*, 19:214–230, 1993. The drawing package dot is available from <http://www.research.att.com/sw/tools/graphviz/>.
- [23] The AGD-Library: Algorithms for Graph Drawing, 1998. Available from <http://www.mpi-sb.mpg.de/~mutzel/dfgdraw/agdlib.html>.
- [24] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1999.
- [25] Debabrata Ghosh, Nevin Kapur, Justin E. Harlow III, and Franc Brglez. Synthesis of Wiring Signature-Invariant Equivalence Class Circuit Mutants and Applications to Benchmarking. In *Proceedings, Design Automation and Test in Europe*, pages 656–663, Feb 1998. Also available at <http://www.cbl.ncsu.edu/publications/#1998-DATE-Ghosh>.
- [26] K. Kozminski, (Ed.). OASIS2.0 User’s Guide. MCNC, Research Triangle Park, N.C. 27709, 1992. (Over 600 pages, distributed to over 60 teaching and research universities worldwide).
- [27] VPR and VPACK: Versatile Packing, Placement and Routing for FPGAs, 1998. Available from <http://www.eecg.toronto.edu/~vaughn/vpr/vpr.html>.
- [28] M. Stallmann, F. Brglez, and D. Ghosh. Heuristics and Experimental Design for Bigraph Crossing Number Minimization. In *ALENEX’99, the First Workshop on Algorithm Engineering and Experimentation*, January 1999. Also available from <http://www.cbl.ncsu.edu/publications/#1999-ALENEX-Stallmann>.
- [29] D. M. Schuler and E. G. Ulrich. Clustering and Linear Placement. In *Proceedings of the 9th Design Automation Workshop*, pages 50–56, 1972.
- [30] Sungho Kang. Linear ordering and Application to Placement. In *Proceedings of the 20th Design Automation Conference*, pages 457–464, 1983.
- [31] A. Hashimoto and J. Stevens. Wire Routing by Optimizing Channel Assignment within Large Apertures. In *Proceedings of the 8th Design Automation Workshop*, pages 155–169, 1971.
- [32] J. E. Harlow and F. Brglez. Design of Experiments in BDD Variable Ordering: Lessons Learned. In *Proceedings of the International Conference on Computer Aided Design*. ACM, November 1998. Also available from <http://www.cbl.ncsu.edu/publications/#1998-ICCAD-Harlow>.
- [33] F. Brglez. Design of Experiments to Evaluate CAD Algorithms: Which Improvements Are Due to Improved Heuristic and Which Are Merely Due to Chance? Technical Report 1998-TR@CBL-04-Brglez, CBL, CS Dept., NCSU, Box 7550, Raleigh, NC 27695, April 1998. Also available at <http://www.cbl.ncsu.edu/publications/#1998-TR@CBL-04-Brglez>.
- [34] J. E. Harlow and F. Brglez. Design of Experiments for Evaluation of BDD Packages Using Controlled Circuit Mutations. In *Proceedings of the International Conference on Formal Methods in Computer-Aided Design (FMCAD’98)*. Springer Verlag, Lecture Notes in Computer Science, November 1998. Also available from <http://www.cbl.ncsu.edu/publications/#1998-FMCAD-Harlow>.
- [35] S. Yang. Logic Synthesis and Optimization Benchmarks User Guide. Technical Report 1991-IWLS-UG-Saeyang, MCNC, Research Triangle Park, NC, January 1991. Now available from <http://www.cbl.ncsu.edu/publications/#1991-IWLS-UG-Saeyang> and benchmarks from <http://www.cbl.ncsu.edu/benchmarks/Benchmarks-upto-1996.html>.