

Graph Mutants based on Characteristic Spanning Trees: Experimental Design of Heuristics for NP-Hard Problems

Debabrata Ghosh Franc Brglez Mathias Stallmann

September 1999

1999-TR@CBL-03-Brglez

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

©1999 authors and CBL, All Rights Reserved

ABOUT THIS DOCUMENT

Abstract – *This paper introduces a comprehensive approach to the generation of random instances of graphs relevant to NP-hard problems in circuit design. Good experimental design technique dictates that treatments (heuristics) be applied to a class of experimental subjects (graphs, circuits) that share clearly identifiable common characteristics. Relevance is achieved only if these characteristics relate to the performance of heuristics on real circuits. We demonstrate that a characteristic spanning tree class of a reference graph can be used to generate random graphs whose behavior w.r.t. various treatments is like that of the reference graph. Results for both crossing number and hypergraph partitioning are discussed.*

Keywords: graph equivalence classes, benchmarking, crossing number, graph partitioning, experimental design of heuristics for NP-hard problems.

Note. This document has been published for viewing on the Web in pdf, 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{
  1999-TR@CBL-03-Ghosh,
  author = "D. Ghosh and F. Brglez and M. F. M. Stallmann",
  title = "{Graph Mutants based on Characteristic Spanning Trees:
           Experimental Design of Heuristics for NP-Hard Problems}",
  institution = "{CBL, CS Dept., NCSU, Box 8206,
                 Raleigh, NC 27695}",
  number = "1999-TR@CBL-03-Ghosh",
  month = "Sep",
  year = "1999",
}
```

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

CONTENTS

I	Introduction	2
II	Background and Motivation	3
III	Bigraph Characterization	5
IV	Bigraph Equivalence Class Synthesis	6
V	Netlist Equivalence Class Synthesis	7
VI	Experimental Results	10
VII	Conclusions	11

LIST OF FIGURES

1	Illustrating an experimental design methodology using three algorithms as <i>treatments</i> and sampling three populations of netlist equivalence classes, each derived from a single instance of a known reference netlist.	4
2	Illustrative instances of small <i>single connected component</i> graphs with the same basic parameters: 17/10 nodes at two layers and 50 edges. Each graph induces an isomorphism equivalence class that presents a very obvious challenge when minimizing the crossing number of each class representation. The only exception is the case of treatment TR17 when applied to the isomorphism class generated from the graph instance under (b).	7
3	This experiment illustrates the fundamental difficulty of generating a class of 2-layer graphs where each instance presents a problem that is comparable to the problem of minimizing the crossing number in all instances of graphs drawn from the isomorphism class. The two reference graphs that form the two very different isomorphism classes are differentiated by the average crossing number of the underlying spanning tree class – a characteristic of the reference circuit. Note that none of the four approaches investigated in this experiment can generate two distributions that are similar to the characteristic distributions of <i>both</i> isomorphism classes. See Figure 5 to assess the similarity of the equivalence class generated with a method proposed in this paper to the isomorphism class in this Figure.	8
4	The average crossing number of the isomorphism graph class is related to the average crossing number of the spanning trees induced by the underlying reference graph — provided we can find an ordering for each spanning tree that truly minimizes the crossing number of the tree.	9
5	This experiment contrast the isomorphism classes of 2-layer graphs with two mutant classes generated with a method proposed in this paper. The isomorphism classes are characterized with two very distinctive crossing number distributions. In contrast to equivalence classes analyzed in Figure 3, the new mutant equivalence classes are clearly very similar to the respective isomorphism classes (in terms of the distributions of the crossing number).	9
6	This experiment contrast the isomorphism classes of multi-level netlists with the equivalence classes of netlist mutants, generated with a method proposed in this paper. In contrast to equivalence classes analyzed in Figure 1, the new mutant equivalence class is clearly very similar to the respective isomorphism class (in terms of the distributions of mincut as well as the crossing number).	11
7	Extended summary of experimental results with four equivalence classes based on three ‘difficult’ reference netlists and three algorithms, reporting either the size of the mincut or the crossing number. Notably, only the equivalence class labeled as ‘mut2’ and described in this paper is consistently similar to the underlying isomorphism class.	12

Graph Mutants based on Characteristic Spanning Trees: Experimental Design of Heuristics for NP-Hard Problems

Debabrata Ghosh

Franc Brglez

Matthias Stallmann

Dept. of Computer Science, Box 8206
NC State University, Raleigh, NC 27695, USA
<http://www.cbl.ncsu.edu/experiments/>

Abstract – *This paper introduces a comprehensive approach to the generation of random instances of graphs relevant to NP-hard problems in circuit design. Good experimental design technique dictates that treatments (heuristics) be applied to a class of experimental subjects (graphs, circuits) that share clearly identifiable common characteristics. Relevance is achieved only if these characteristics relate to the performance of heuristics on real circuits. We demonstrate that a characteristic spanning tree class of a reference graph can be used to generate random graphs whose behavior w.r.t. various treatments is like that of the reference graph. Results for both crossing number and hypergraph partitioning are discussed.*

Keywords: graph equivalence classes, benchmarking, crossing number, graph partitioning, experimental design of heuristics for NP-hard problems.

I. INTRODUCTION

Polynomial-time heuristics, devised to solve NP-hard problems, generally provide no guarantee as to the optimality of the solution. Changing the starting point for the problem instance can induce unpredictable variability of results when experiments are repeated. To systematically study the behavior of such heuristics, we advocate the fundamental principles of experimental design, *randomization*, *replication*, and *organization to reduce error*, first formalized by R. A. Fisher in the 1920s for analyzing problems in agriculture and medicine [1]. The number of books on the subject of experimental design today is overwhelming. In this work, we adapt the models and the methods from [2].

An experimental design involves three basic steps: (1) selection of an equivalence class of experimental subjects, eligible for treatments, (2) application of one or more treatment to the same class, and (3) statistical evaluation of each treatment effectiveness. In the context of experimental evaluation of graph-

based algorithms, these principles are embodied in the creation of a *graph equivalence classes*, and repetition of the experiments for each member in the class. In our work, we consider ‘treatments’ as algorithms applied to such equivalence classes.

The traditional approach to comparing the performance of two or more algorithms relies on graph equivalence classes which are generated as random instances under relatively loose constraints, e.g. the random graphs used to evaluate partitioning, coloring, and TSP algorithms in [3, 4, 5], the generators in Stanford GraphBase [6] used to evaluate the crossing number minimization algorithms [7], etc.

The exclusive use of random graphs to compare the performance of algorithms has drawbacks: we may fail to detect the difference in performance of two algorithms when one exists; we may bias the design of an algorithm for data sets that will seldom if ever be encountered in applications where a specific cost function minimization is truly important. For example, a placement algorithm optimized for a class of random graphs may not necessarily do as good a placement and wire crossing minimization of relatively sparsely interconnected modules arising in a complex VLSI circuit design. On the other hand, the problem of generating equivalence classes of graphs that are representative of graphs representing realistic design cases is hard, and is only beginning to be addressed [8, 9, 10, 11, 12, 13, 14, 15].

The graph classes considered in this paper are abstractions of design- and domain-specific netlists, i.e. directed hypergraphs. Two netlists, one representing a multiplier-like device, the other a specific controller device, may have the same number of inputs/outputs, the same number, the same size, and the same distributions of cell nodes, and nets (hyperedges), and yet they may differ significantly in the layout area and total wire length and wire crossing after embedding onto a plane and routing all nets under technology-specific design rules. The equiv-

alence class of all netlists with the same I/O, cells and net distributions, while also including instances of random graphs, is clearly too broad for evaluating the performance of any placement and routing algorithms. In this paper we examine and propose solutions to two related problems: (1) how to evaluate differences between two or more equivalence classes, and (2) how to generate an equivalence class that will retain the most important characteristics of a specific reference netlist such as one representing a multiplier-like device, a controller device, etc.

As a reference point for other graph classes (and representing a counterpoint to unconstrained random classes) is *isomorphism class* of a specific graph, defined more precisely later, but essentially the same graph presented as input in different orders and with different labels.

The paper is organized into several sections as follows:

- *Background and Motivation*, introducing experimental design methodology, adopting three algorithms as *treatments*, and sampling three populations of netlist equivalence classes, each derived from a single instance of a known reference netlist;
- *Bigraph Characterization*, introducing experimental design to characterize differences in 2-layer (i.e. bipartite) graphs with the same number of edges and nodes at each layer;
- *Bigraph Equivalence Class Synthesis*, introducing a new class of bigraph mutants that are different but similar to isomorphism class of its reference bigraph;
- *Netlist Equivalence Class Synthesis*, introducing a new class of netlist mutants that are different but similar to the isomorphism class of its reference netlist;
- *Experimental Results*, demonstrating the effectiveness of the proposed solution to the problem illustrated in Figure 1, along with tabulated results for other known difficult cases of generating equivalence classes of netlists that retain a demonstrated degree of similarity to the respective isomorphism class of the reference netlist.
- *Conclusions*, summarizing the current status and outlining possibilities for future work.

II. BACKGROUND AND MOTIVATION

We use experimental design methodology to illustrate key issues that arise when evaluating the relative performance of two or more algorithms (treatments) using two or more equivalence classes.

The three treatments correspond to repeated executions and evaluations of three algorithms: two al-

gorithms implement balanced netlist bi-partitioning heuristics [17, 18], and one deals with rank-order directed graph placement [19]. The three equivalence classes are the isomorphism class as defined in [15], the class of clones [11], and the class of mutants [13, 16].

Briefly, we create the netlist isomorphism class \mathcal{N}_{iso} as follows: (1) take a reference netlist represented as a hypergraph $G_r(V, E)$, (2) apply, uniformly to all nodes in G_r , a *random re-order*, and *random re-label* procedure $rr(V)$:

$$\mathcal{N}_{\text{iso}} = \{N_j \in G_r(rr(V), E)\} \quad (1)$$

Relative to all other instances in \mathcal{N}_{iso} , each instance N_j has the following properties:

P1: the order of nodes in N_j is uniformly random;

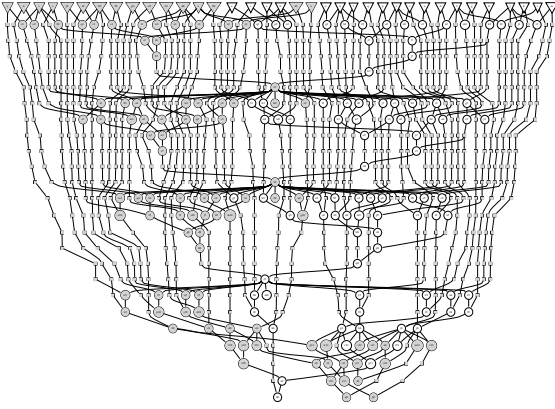
P2: the labels of nodes in N_j are uniformly random.

Properties (*P1*, *P2*) are essential to good experimental design and must be maintained universally for all equivalence classes, not just the isomorphism class. The purpose of *P1* is clear. Without *P2*, programs that rely on hashing input data may *unknowingly undo* the randomization of input presentations and thus confound the experiments. No assumptions should be made on how and whether an algorithm changes the order of the input data read into memory: important lessons about the consequences of making such assumptions have been learned and are reported in [20]. We shall further underscore the importance of this class with the experimental results that will be discussed later in this and subsequent sections.

The class of clones [11] relies on a specific netlist characterization which must be satisfied to form an equivalence class of related ‘clone circuits’. The class of mutants [13, 16] differs from the class of clones by way of the netlist characterization and the constraints that must be satisfied when generating a ‘mutant circuit’.

Nominally, each instance of a netlist from *any* equivalence class is related to a *reference netlist* that may typically represent an application-specific design rather than a netlist generated randomly. An example of a small reference netlist (c432) is shown in Figure 1-a; it is based on a 36-input/7-output logic circuit with 179 2-input cell nodes distributed over 24 logic levels. This netlist implements the function of a priority decoder and is one of the smallest that have been used as a netlist benchmark since 1985 [21]. The netlist has been converted to a *k-partite canonical graph form* and drawn with the *dot* program [19].

(a) A small reference netlist (C432), drawn in a canonical graph form.



- The canonical graph form is a k-partite graph induced by a netlist (a hypergraph) as defined in [13, 16].
- Shaded/unshaded nodes belong to balanced bi-partitions connected by 11 nets (hyperedges).

(b) Results summary of an experimental design based on three treatments and three equivalence classes of netlists.

The three treatments correspond to repeated executions and evaluations of three algorithms: two algorithms implement balanced netlist bi-partitioning heuristics [17, 18], and one deals with rank-order directed graph placement [19]. The three equivalence classes are the isomorphism class as defined in [15], the class of clones [11], and the class of mutants [13, 16]. This summary of an experimental design illustrates the key issue when testing the performance of these algorithms, each implementing a heuristic solution to a well-known NP-hard problem:

Which of these classes, if any, is best suited to evaluate the comparative performance of each algorithm?

Finding a new class of netlists that would appear *very similar* to the isomorphism class, and yet be all different, is the challenge addressed in this paper. See Figure 6 to assess the similarity of the equivalence class generated with a method proposed in this paper to the isomorphism class in this Figure.

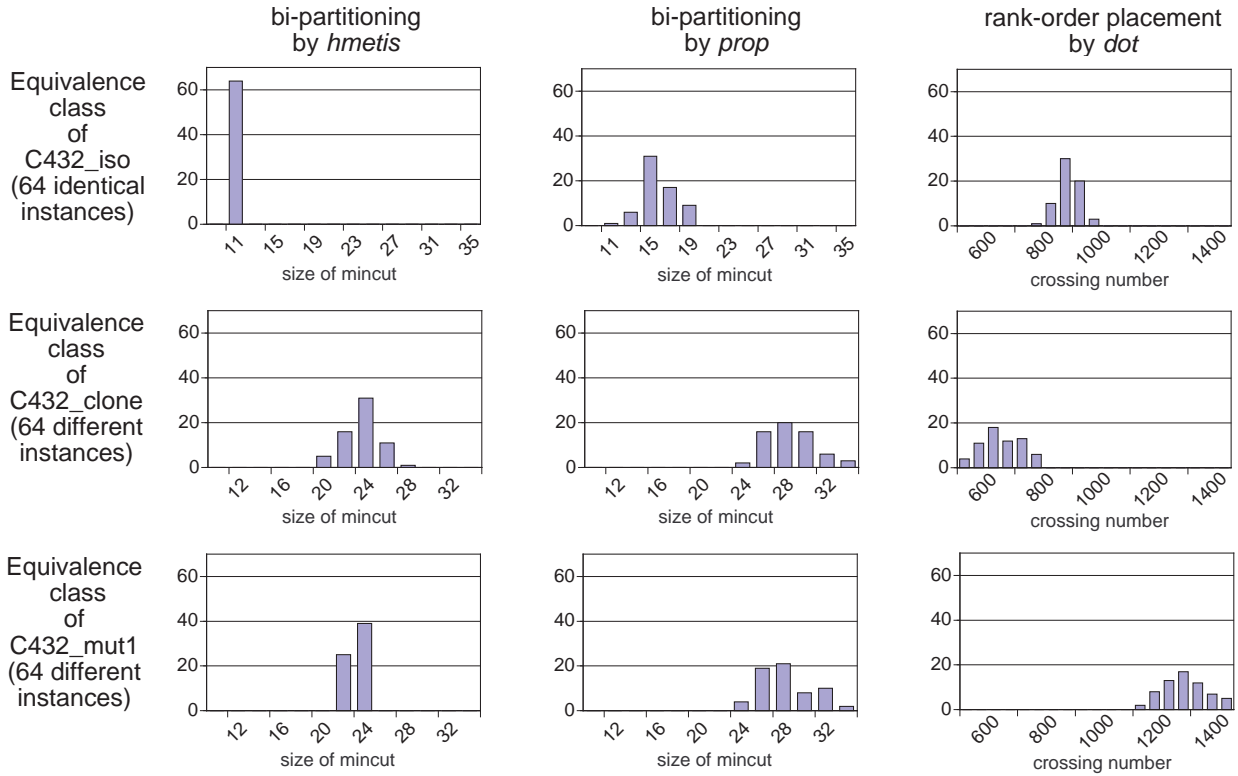


Fig. 1. Illustrating an experimental design methodology using three algorithms as *treatments* and sampling three populations of netlist equivalence classes, each derived from a single instance of a known reference netlist.

This form is a simple transformation of the underlying netlist; it is a rank-order structure of alternating *net nodes* and *cell nodes*. The form also defines the netlist signature and the corresponding equivalence class of mutants [13, 16].

A statistical summary of the proposed experimental design is shown in Figure 1-b. Three algorithms are applied to all 64 netlist instances in each class: a partitioner `hmetis` [17], a partitioner `prop` [18], and a placer `dot` [19]. Both partitioners report the number of nets (hyperedges) cut in a balanced bipartition. The placer reports the total of all edge crossings after optimizing the placement of all net nodes and all cell nodes in the k-partite graph form of the netlist. The results reported here are for illustration only. Equivalence classes based on additional reference circuits should be considered if the objective is to quantify relative performance of two algorithms. Our observations about each equivalence class follow the figure from the top down:

Isomorphism class: If each algorithm could solve the problem optimally, all results reported for mincut and the crossing number would have a variance of 0! The fact that only `hmetis` returns a mincut of 11 for all 64 instances in this class does not imply its optimality — it just appears to perform better than `prop` for *this* equivalence class. The distributions, with relatively large variance, observed for `prop` and `dot` are what may be observed typically, especially when the size of the circuit increases.

Class of clones: The centers of distributions induced by this class in terms of the three algorithms have clearly shifted significantly w.r.t. the distributions of the isomorphism class. In fact, one can argue that the netlist in this class are far from similar to the isomorphism class of the reference netlist. Moreover, while the centers of the mincut distribution shifted to the right, the center of the crossing number distribution shifted to the left.

Class of mutants: The centers of distributions induced by this class in terms of the three algorithms have also shifted significantly w.r.t. the distributions of the isomorphism class — in this case all to the right. Again, one can argue that the netlist in this class are far from similar to the isomorphism class of the reference netlist.

A number of additional treatments (heuristics) will be used to evaluate the equivalence classes throughout this paper. These treatments have been assigned a ‘serial number’ in an earlier paper [22]. For example, TR03 refers to conventional `barycenter` heuristic [23], TR12 refers to `dot` heuristic [19], and TR17

refers to the new heuristic introduced in [22] and refined in [24]. The rhetorical question at this point is:

Which of these classes, if any, is best suited to evaluate the comparative performance of each algorithm?

We argue that the isomorphism class is clearly one that exposes a number of problem areas with a design of a heuristic that is being evaluated. Neither the clone class nor the mutant class as implemented at present are representative (in the cases shown) of the underlying isomorphism class. We may learn more about the limitations of the heuristic by looking for another reference circuit and create an isomorphism class with the center of distribution shifted from the first class. However, finding an equivalence class of *different netlists* but a distribution that is similar to the underlying isomorphism class of the reference circuit has a number of merits such as (1) increased diversity of subjects, but within well-defined bounds, and (2) increased number of tests by creating additional isomorphism classes of each distinct subject.

We pursue this goal in several stages, expanding and refining the netlist mutation process we have tested to date. The next two sections revisit the basic primitive we use throughout the procedure: the single *connected component* of the 2-layer graph (bigraph) and examine critically a number of ways to mutate this component alone. Only then we revisit the process of mutant generation across all level of the k-partite canonical form graph and demonstrate the relative merits of the new procedures.

III. BIGRAPH CHARACTERIZATION

A bigraph class can be characterized by three different parameters that are relevant in the study of crossing numbers. In place of the usual $n_0 =$ number of layer-0 nodes, $n_1 =$ number of layer-1 nodes, and $m =$ number of edges, we look at a *basic signature* (a, b, m) , where $a = m / (n_0 + n_1 - 1)$ (total edges/spanning-tree edges), $b = n_0 / n_1$ (balance factor), and m . The reasoning behind this form becomes evident later.

This is far from sufficient, however, if we want meaningful experimental results that are applicable to subjects in a given class. Consider, for example, the bigraphs shown in Figure 2. All belong to the class with basic signature $(\frac{25}{13}, 1.7, 50)$, i.e. $n_0 = 17$, $n_1 = 10$, and $m = 50$ and are randomly generated in different ways.

The graph in (a) is generated using standard techniques for generating random bigraphs (see, e.g. [6]). The degree of each node on either layer is a random

variable. The graphs in (b) and (c) have a fixed degree of 5 on layer 1 (typical of cell nodes in VLSI circuits) and random degree on layer 0. In addition, they were generated from spanning trees that have two different crossing numbers (via heuristic TR17 [22], which is known to do very well on trees [24]).

Each of the three graphs was used to generate a 64-subject isomorphism class and two heuristics were applied to each class, TR12, the `dot` heuristic [19], and TR17 from [22]. Which heuristic is better? Results on the isomorphism class of graph (a) are inconclusive. For graph (b) TR17 appears to find an optimal solution consistently while TR12 has a large variance (but still finds the optimal solution occasionally). Graph (c) again leads to inconclusive results with both heuristics showing a large variance (and neither achieving an optimal solution). The answer to the question clearly depends on graph characteristics beyond the basic signature. For graphs that are like graph (b) TR17 is the better choice. For those like graph (c) we may want to look for better heuristics. And graphs generated randomly without constraints like graph (a) make it difficult to draw any conclusions. In fact, as m gets larger while a and b remain the same, they are likely to consist of multiple connected components and the basic signature becomes meaningless. See [24] for further experimental evidence that unconstrained random graphs of the same signature, even when connectivity is maintained, make poor experimental subjects when it comes to distinguishing the performance of heuristics.

IV. BIGRAPH EQUIVALENCE CLASS SYNTHESIS

We turn now to the problem of creating classes based on particular reference graphs. For illustration purposes we have chosen two reference graphs with the same basic signature $(2.0, 1.5, 1910)$ but widely different crossing numbers (as reported by TR17). The first row of Figure 3 illustrates the performance of TR17 on the two isomorphism classes.

The remaining rows show various attempts to synthesize reasonable random mutation classes. Graph classes having only the same basic signature as the reference graphs are shown in rows 2 and 3. In each case a random spanning tree of the reference graph was generated and then augmented with randomly chosen extra edges to fulfill the signature. The only difference between the two methods is that the extra edges are purely random in row two constrained by *windowing* in row 3. Roughly, windowing means that cell nodes are more likely to be connected to net nodes “close” to them in a given ordering of the

graph. Imagine the nodes of each layer being equally spaced so that the two layers have the same width. Each cell node is then more likely to be connected to the net nodes with the closest x -coordinates. One windowing method is described in more detail later. In row 3 the windowing was based on a random ordering of the nodes on each layer.

Random graphs with only the same basic signature have much higher crossing numbers than either reference graph. Windowing reduces the crossing number somewhat, but, based as it is on random ordering, not significantly. The original reference graph used makes no noticeable difference in the results.

The last two rows show results for classes based on *extended signature* — along with the basic signature the degree sequence of the cell nodes is maintained. In this case all cell nodes of each reference graph have degree 5 and this fact is preserved as extra edges are added to the spanning tree. Without windowing (row 4) the number of crossings is large for both reference graphs, similar to the corresponding results for basic signature (row 2). The bottom row shows results for extended signature with windowing, but in this case windowing is used during the construction of the spanning tree as well as when extra edges are added. The fact that windowing is integrated throughout the random generation of the graph accounts for the dramatic drop in crossing number. Again, neither construction shows significant difference in behavior that can be attributed to the reference graph.

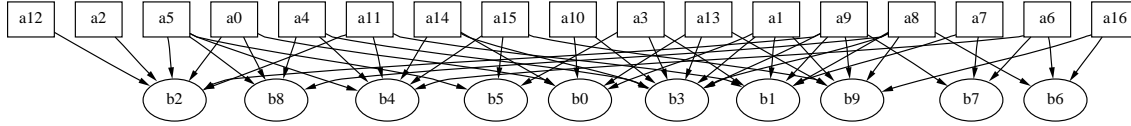
This leaves us with a fundamental question: how can we distinguish between graphs that have the same basic and extended signatures but different crossing numbers and can the distinction be used to generate random graphs with similar characteristics?

Recently we discovered that classes of trees — basic signature $(1, r, m)$ — exhibit the most radical differences in crossing number. Perhaps the underlying structure of a graph (at least w.r.t. crossing number) can be characterized by its spanning trees. Figure 4 illustrates the results of an experiment along these lines. For reference graphs having basic signatures $(4, 0.25, m)$ and $(2, 1.5, m)$ with m ranging from 230 to 1910 we created isomorphism classes and *characteristic spanning tree* classes. The latter were generated by selecting 64 random spanning trees of each reference graph (using a minimum-spanning-tree algorithm with random weights assigned to the edges of the graph). The figure shows that (using TR17) there is a relationship between the average crossing number of an isomorphism class and that of the characteristic spanning tree class. The exact nature of the relationship is still being investigated but it appears

(a) A typical graph instance generated randomly, e.g. by way of GraphBase [6].

Here, the degree of node at each layer is a random variable. We report the crossing numbers of 64 graph instances from the isomorphism class induced by this graph for two treatments: TR12 is based on the heuristic in [19]. TR17 is based the heuristic in [22, 24].

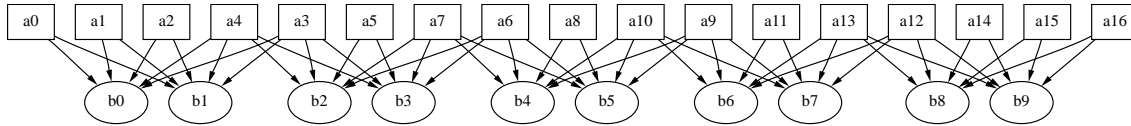
treatment	min	avg	max
TR12	186	205.6	222
TR17	208	209.0	216



(b) A graph instance generated from a spanning tree class with the crossing number average of 4.0 (G_{-10}).

Here, only the degree of node at layer 0 is a random variable. The nodes at layer 1 all have degree 5. The characteristic spanning tree class of a particular reference graph was used to generate this graph instance. As above, 64 instances of the isomorphism class were used.

treatment	min	avg	max
TR12	66	101.6	186
TR17	66	66.0	66



(c) A graph instance generated from a spanning tree class with the crossing number average of 6.7 (G_{-11}).

As with (b), degree at layer 0 is random, degree at layer 1 is fixed (= 5), and a characteristic spanning tree class was used, but from a different reference graph.

treatment	min	avg	max
TR12	127	132.4	201
TR17	127	129.5	266

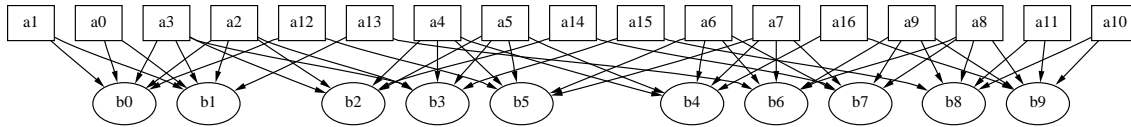


Fig. 2. Illustrative instances of small *single connected component* graphs with the same basic parameters: 17/10 nodes at two layers and 50 edges. Each graph induces an isomorphism equivalence class that presents a very obvious challenge when minimizing the crossing number of each class representation. The only exception is the case of treatment TR17 when applied to the isomorphism class generated from the graph instance under (b).

to depend on the first two parameters of our basic signature. It is certainly clear that reference graphs with smaller crossing numbers lead to characteristic spanning tree classes with smaller average crossing numbers.

We are now ready to present a new method of generating a class of random mutants from a reference graph G . First a random spanning tree T of G is chosen. The tree T is then embedded using TR17 so as to minimize the number of crossings (theoretically we could use an exact algorithm since the crossing number problem can be solved in polynomial time for trees, albeit via a complicated algorithm [25]). The resulting ordering becomes the basis for a windowed approach to adding extra edges randomly. Figure 5 shows that the new mutant class *mut2* does a re-

markably good job of replicating the crossing number distributions for the isomorphism classes of the two reference graphs used in Figure 3. For the reference graph G_{-7-10} we were tempted to suspect that the subjects in the mutant class were all isomorphic to the reference graph. Not so. The 64 random mutants used to obtain the singular distribution in the lower left histogram exhibited 53 distinct degree sequences!

V. NETLIST EQUIVALENCE CLASS SYNTHESIS

An electrical netlist is modeled as a $2k$ -partite graph¹ where k = number of (logic) levels in the circuit. If layers are numbered $0, \dots, 2k - 1$ (from the top down

¹The netlist, which is a hypergraph, is converted to a multi-level, $2k$ -partite graph through a series of transformations including insertion of feedthrough nodes [13].

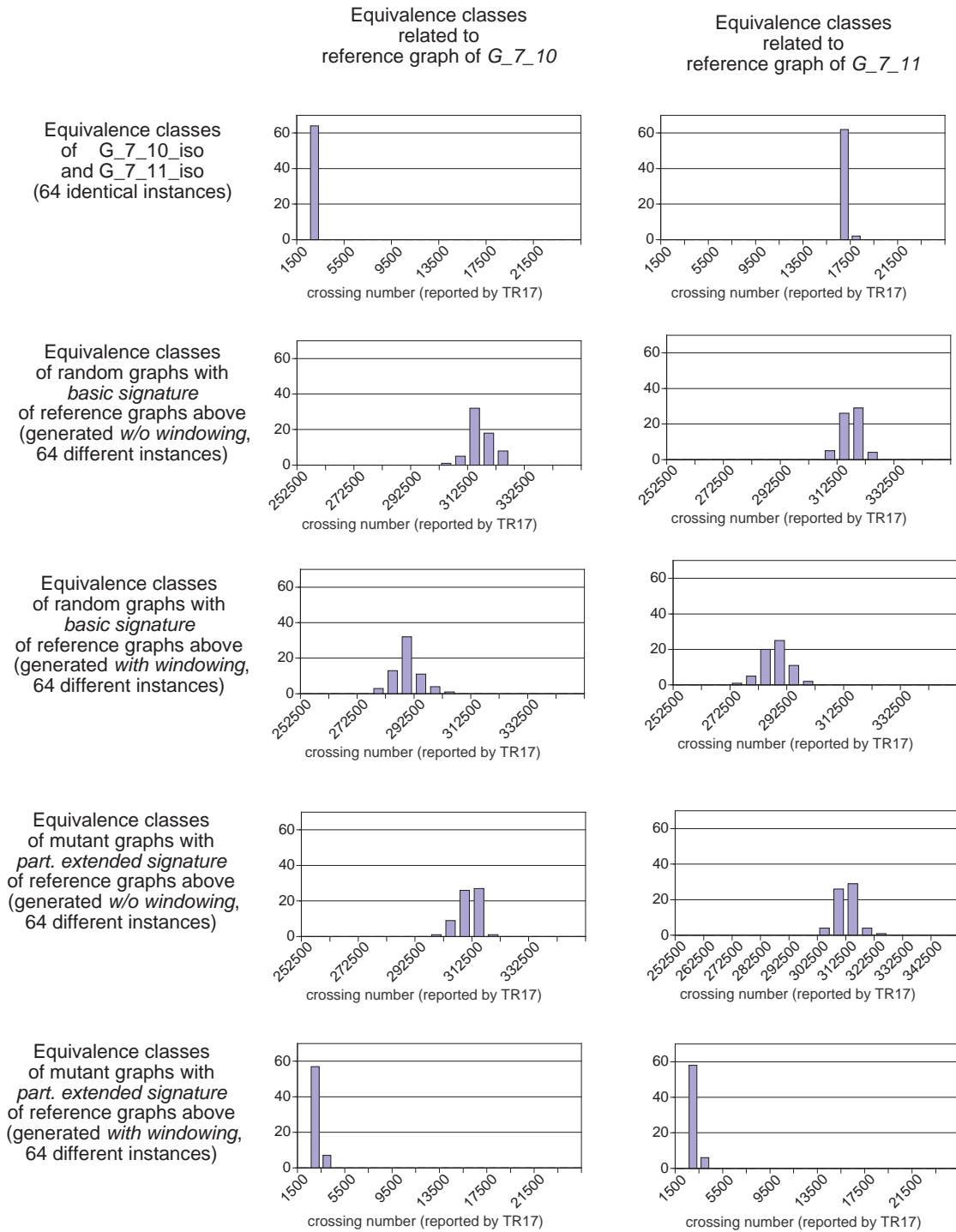
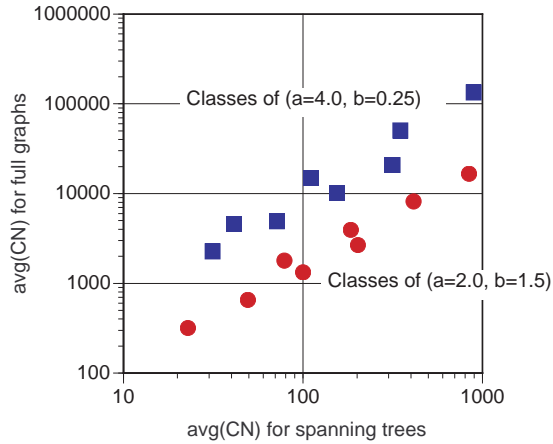


Fig. 3. This experiment illustrates the fundamental difficulty of generating a class of 2-layer graphs where each instance presents a problem that is comparable to the problem of minimizing the crossing number in all instances of graphs drawn from the isomorphism class. The two reference graphs that form the two very different isomorphism classes are differentiated by the average crossing number of the underlying spanning tree class – a characteristic of the reference circuit. Note that none of the four approaches investigated in this experiment can generate two distributions that are similar to the characteristic distributions of *both* isomorphism classes. See Figure 5 to assess the similarity of the equivalence class generated with a method proposed in this paper to the isomorphism class in this Figure.



The goal of the experiments summarized in this figure is to demonstrate the level of correlation between the average crossing number of the characteristic spanning tree class of a given 2-layer reference graph and the average crossing number we evaluate for the isomorphism class induced by the same reference graph.

The experiments were conducted for a total of 4×4 reference graphs, with 4 graphs in a group where only the number of edges doubled for each reference graph; ratios such as $a = \text{total_edges}/\text{spanning_tree_edges}$ and $b = \text{nodes@lev0}/\text{nodes@lev1}$ remained constant. These families of graphs are defined in more detail in [24]. We see here that *the average crossing number of the isomorphism graph class is related to the average crossing number of the spanning trees induced by the underlying reference graph*. This property allows us to synthesize an equivalence class of 2-layer mutant graphs that are 'similar' to the isomorphism class induced by the the same reference graph.

The results shown are for crossing number averages as reported by treatment TR17 [22, 24]. Such correlations cannot be demonstrated for any other algorithm currently available.

Fig. 4. The average crossing number of the isomorphism graph class is related to the average crossing number of the spanning trees induced by the underlying reference graph — provided we can find an ordering for each spanning tree that truly minimizes the crossing number of the tree.

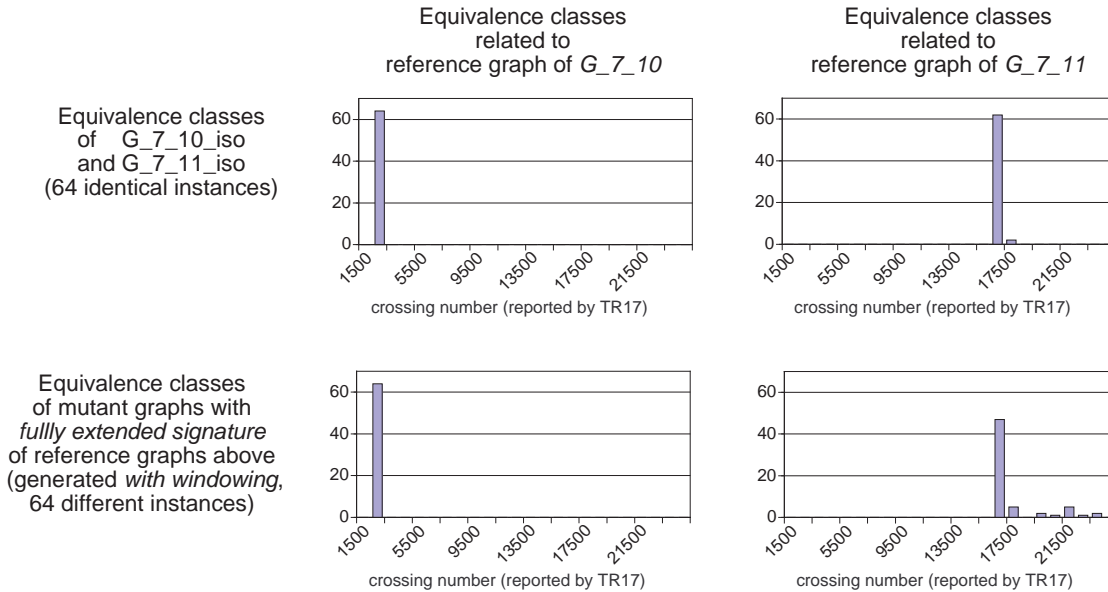


Fig. 5. This experiment contrast the isomorphism classes of 2-layer graphs with two mutant classes generated with a method proposed in this paper. The isomorphism classes are characterized with two very distinctive crossing number distributions. In contrast to equivalence classes analyzed in Figure 3, the new mutant equivalence classes are clearly very similar to the respective isomorphism classes (in terms of the distributions of the crossing number).

in Figure 1(a)) the even layers contain net nodes and odd layers contain cell nodes. Connections from layer $2i - 1$ to $2i$ (for $1 \leq i \leq k - 1$) are one-to-one — each cell connects to single net belonging to it. A *slice* is the bigraph induced by layers $2i$ and $2i + 1$ for some $i < k$ (identify this as slice i). A *local connected component (LCC)* is a connected component of a slice. The LCC's induce a k -partite graph called the *CC-graph* with an edge between an LCC C_{i-1} in slice $i-1$ and C_i in slice i whenever C_{i-1} has a cell connected to a net of C_i .

Our new algorithm for synthesizing a mutant class from a netlist is a modification of an earlier one that has worked well in several contexts [16]. The original algorithm works roughly as follows:

```

for each slice  $i = 0, \dots, k-1$  do
  for each LCC  $C$  in slice  $i$  do
    generate a mutant for  $C$ 
    (using windowing)

```

Two ideas emerge in the new algorithm, both based on the notion that the order of nodes on a layer is important if windowing is to succeed. First, the CC-graph is placed using the *dot* heuristic, the resulting placement imposing an ordering on the LCC's in each slice. Second, the ordering of cell nodes in an LCC of C slice $i > 0$ is derived from slice $i - 1$: the net nodes of C are ordered according to the LCC's of slice $i - 1$ that are connected to C in the CC-graph (recall the one-to-one correspondence) and the cell nodes of C are ordered using a single pass of the barycenter heuristic on a random spanning tree of the LCC). Slice 0 is a special case. Since no ordering is imposed by a previous slice, TR17 is used on the spanning tree of each LCC to get an ordering of minimal crossing number before adding extra edges via windowing. To summarize:

```

find the CC-graph and order LCC's within
  each slice using dot
for each LCC  $C$  in slice 0 do
  generate a random spanning tree of  $C$ 
  use TR17 to order the nodes of  $C$ 
  add extra edges using windowing
for each slice  $i = 1, \dots, k-1$  do
  for each LCC  $C$  in slice  $i$  do
    generate a random spanning tree  $T$ 
      of  $C$ 
    order the net nodes of  $C$  from the
      cell nodes of LCC's
      in slice  $i-1$  connected to  $C$ 
    order the cell nodes of  $C$  using a
      single pass of the
      barycenter heuristic
    add extra edges using windowing

```

VI. EXPERIMENTAL RESULTS

This section completes the experimental design introduced in Figure 1 earlier. Again, we apply the three algorithms to all 64 netlist instances of the new mutation class *mut2* introduced in this paper : a partitioner *hmetis* [17], a partitioner *prop* [18], and a placer *dot* [19]. The resulting distributions for the new class *c432_mut2* are shown in Figure 6. Clearly, the distributions are very similar to the distributions for the underlying isomorphism classes of the reference netlists.

To demonstrate that the improvement is not an isolated instance, we also designed similar experiments with additional reference netlists. The criterion we used for the choice of the netlist is the same as the one we applied to *c432*. The netlists *c880* and *c6288* were chosen since we could not find an equivalence class with earlier methods that would be sufficiently similar to the respective isomorphism class. Both of these netlists are also from the ISCAS'85 data set [21]. The netlist of *c6288* is a model of a 16x16 bit array multiplier. As the results in Figure V demonstrate, this reference circuit is a particular challenge for the clone equivalence class. Both the average mincut and the average crossing number of the clone netlists are several orders of magnitude larger than the mincut and the crossing number of the reference netlist.

On the other hand, Figure 6 shows that the mean and the average of both new mutant classes, *c880_mut2* and *c6288_mut2*, have averages as well as min/max values that are very comparable to the ones shown for the respective isomorphism classes.

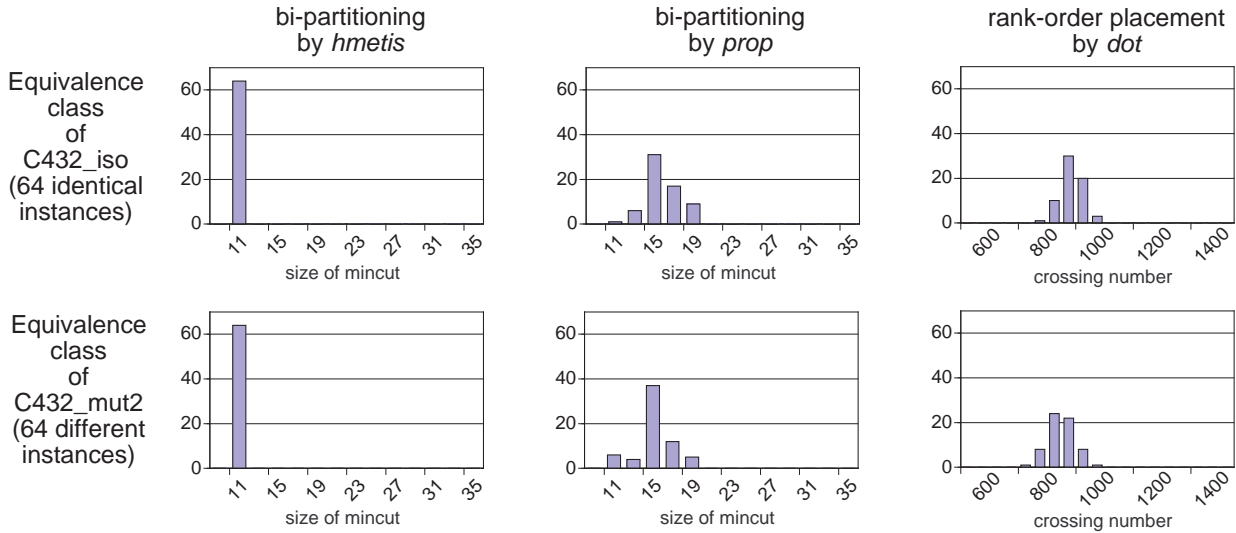


Fig. 6. This experiment contrast the isomorphism classes of multi-level netlists with the equivalence classes of netlist mutants, generated with a method proposed in this paper. In contrast to equivalence classes analyzed in Figure 1, the new mutant equivalence class is clearly very similar to the respective isomorphism class (in terms of the distributions of mincut as well as the crossing number).

VII. CONCLUSIONS

We have successfully applied the fundamental principles of experimental design to demonstrate effective solutions to two related problems: (1) the synthesis of tightly controlled equivalence classes of graph mutants and (2) the methodology of evaluating, with statistical significance, the differences among two or more heuristics (treatments) for a given equivalence class as well as the difference between several equivalence classes for a given treatment.

We have introduced the concept of a *characteristic spanning tree* class of connected components in bi-graphs and used them to generate closely related random bigraphs as the primitive components of large netlists in an equivalence class. This class is similar to and complements the role of the reference netlist isomorphism class in the experimental design of heuristics for NP-hard problems.

Some directions for future work include: (1) more experiments with additional types of heuristics, to gain additional insight about characterization of mutant classes, and (2) testing the scalability of the mutant generation process with larger reference graphs and exploring alternative approaches to generating such classes.

REFERENCES

- [1] R. A. Fisher. *Statistical Methods, Experimental Design, and Scientific Inference*. Oxford University Press, 1993.
- [2] G. E. P. Box, W. G. Hunter, and J. S. Hunter. *Statistics for Experimenters: An Introduction to Design, Data Analysis, and Model Building*. John Wiley & Sons, 1978.
- [3] D. S. Johnson, C. R. Aragon, L. A. McGeogh, and C. Schevon. Optimization by simulated annealing: An experimental evaluation, part I (graph partitioning). *Operations Research*, 37(6):865 – 892, 1989.
- [4] D. S. Johnson. Local optimization and the traveling salesman problem. In *Proc. 17th International Colloquium on Automata, Languages, and Programming*, pages 446 – 461, 1990.
- [5] D. S. Johnson, C. R. Aragon, L. A. McGeogh, and C. Schevon. Optimization by simulated annealing: An experimental evaluation, part II (graph coloring and number partitioning). *Operations Research*, 39(3):378 – 406, 1991.
- [6] Donald E. Knuth. *The Stanford Graphbase*. Addison Wesley, 1993.
- [7] M. Jünger and P. Mutzel. 2-Layer Straightline Crossing Minimization: Performance of Exact and Heuristic Algorithms. *Journal of Graph Algorithms and Applications (JGAA)*, 1(1):1–25, 1997.
- [8] J. Darnauer and W. Dai. A Method for Generating Random Circuits and its Application to Routability Measurement. In *4th ACM/SIGDA Int'l Symp. on FPGAs, FPGA96*, pages 66–72, February 1996.
- [9] Michael Hutton, J.P. Grossman, J. Rose and D. Corneil. Characterization and Parameterized Random Generation of Digital Circuits. In *Proceedings of the 33rd ACM/IEEE Design Automation Conference*, June 1996.
- [10] Michael Hutton, J.P. Grossman, J. Rose and D. Corneil. Generation of Synthetic Sequential Benchmark Circuits. In *ACM Symposium on FPGAs*, pages 149–155, February 1997. Available from <http://www.eecg.toronto.edu/~pubs/pubs.html>.
- [11] Michael Hutton, J.P. Grossman, J. Rose and D. Corneil. Characterization and Parameterized Random Generation

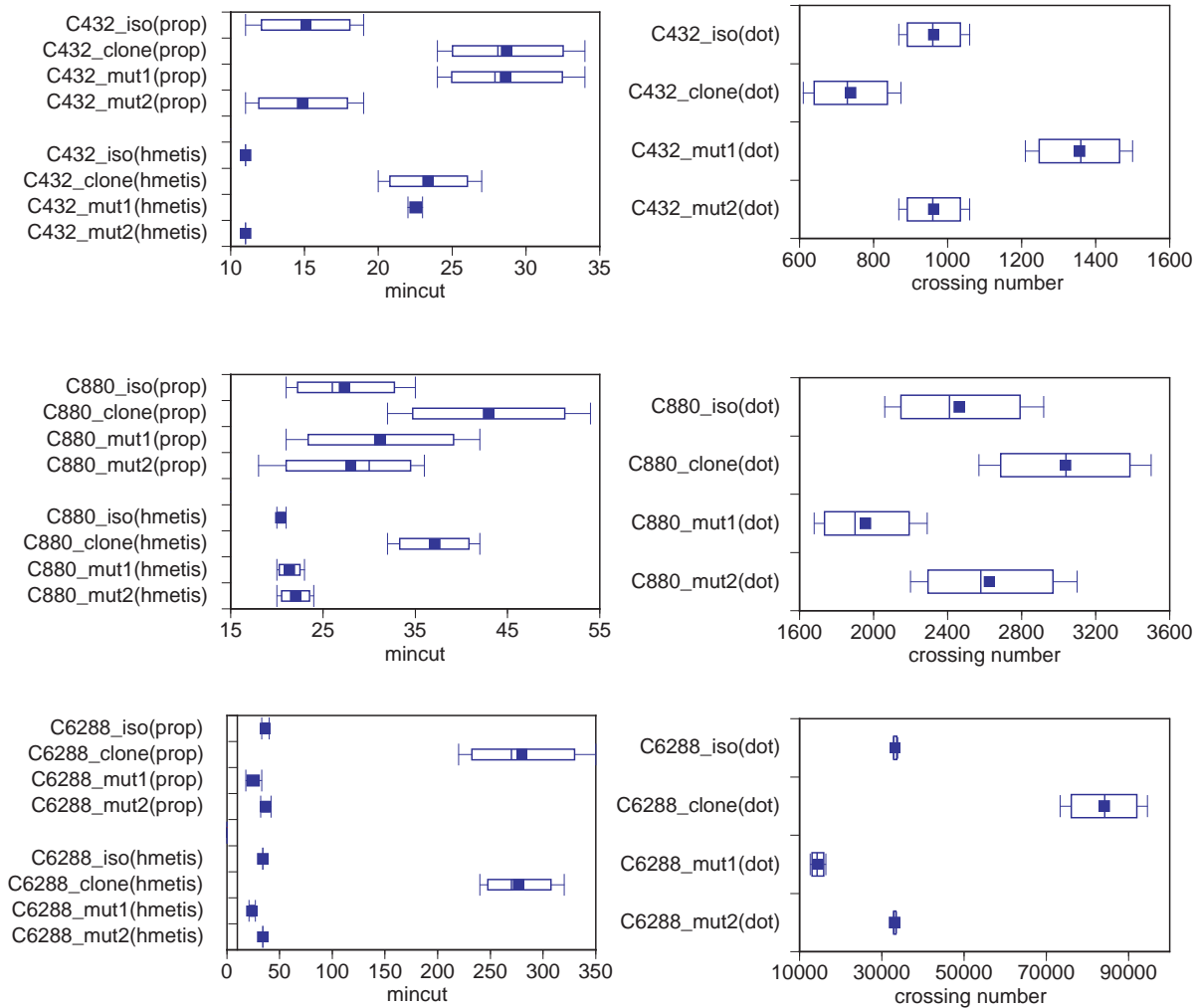


Fig. 7. Extended summary of experimental results with four equivalence classes based on three ‘difficult’ reference netlists and three algorithms, reporting either the size of the mincut or the crossing number. Notably, only the equivalence class labeled as ‘mut2’ and described in this paper is consistently similar to the underlying isomorphism class.

of Combinational Benchmark Circuits. *IEEE Trans. Computer-Aided Design*, 1999. To appear. Postscript available from <http://www.eecg.toronto.edu/~jayar/pubs/pubs.html>. Software available from <http://www.eecg.toronto.edu/~mdhutton/gen/-Terms.html>.

- [12] N. Kapur, D. Ghosh, and F. Brglez. Towards A New Benchmarking Paradigm in EDA: Analysis of Equivalence Class Mutant Circuit Distributions. In *ACM International Symposium on Physical Design*, April 1997.
- [13] 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>.
- [14] J. E. Harlow and F. Brglez. Design of Experi-

ments 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>.

- [15] F. Brglez and R. Drechsler. Design of Experiments in CAD: Context and New Data Sets for ISCAS’99. In *IEEE 1999 International Symposium on Circuits and Systems – ISCAS’99*, May 1999. A reprint is accessible from <http://www.cbl.ncsu.edu/publications/#1999-ISCAS-Brglez>.
- [16] D. Ghosh and F. Brglez. Equivalence classes of circuit mutants for experimental design. In *Proceedings, Intl. Symp. Circuits and Systems (ISCAS)*, May–June 1999. Also available at <http://www.cbl.ncsu.edu/publications/#1999-ISCAS-Ghosh>.

- [17] G. Karypis, R. Aggarwal, V. Kumar, S. Sekhar. Multilevel Hypergraph Partitioning: Application in VLSI Domain. In *34rd Design Automation Conference, ACM/IEEE*, 1997.
- [18] R. Kužnar and F. Brglez. PROP: A Recursive Paradigm for Area-Efficient and Performance Oriented Partitioning of Large FPGA Netlists. In *IEEE International Conference on Computer-Aided Design*, pages 644–649, November 1995.
- [19] 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.
- [20] 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>.
- [21] F. Brglez and H. Fujiwara. Special Session on ATPG (Also introducing 'A Neutral Netlist of 10 Combinational Benchmark Circuits'). In *IEEE 1985 International Symposium on Circuits and Systems - ISCAS85*, 1985. Now a benchmark directory ISCAS85 at <http://www.cbl.ncsu.edu/benchmarks/Benchmarks-upto-1996.html>.
- [22] M. Stallmann, F. Brglez, and D. Ghosh. Heuristics and Experimental Design for Bigraph Crossing Number Minimization. In *Proceedings of the First Workshop on Algorithm Engineering and Experimentation (ALENEX 99)*, January 1999. Also available at <http://www.cbl.ncsu.edu/publications/>.
- [23] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1999.
- [24] M. Stallmann, F. Brglez and D. Ghosh. Heuristics and Experimental Design for Bigraph Crossing Minimization. Technical Report 1999-TR@CBL-04-Stallmann, CBL, CS Dept., NCSU, Box 8206, Raleigh, NC 27695, October 1999. Also available at <http://www.cbl.ncsu.edu/publications/#1999-TR@CBL-04-Stallmann>.
- [25] F. Shahrokhi, O. Sýkora, L. A. Székely, and I. Vrt'o. On bipartite drawings and the linear arrangement problem. In *Lecture Notes in Computer Science*, number 1467, pages 55–68. Springer Verlag, 1997.