Towards A New Benchmarking Paradigm in EDA:
Analysis of Equivalence Class Mutant Circuit Distributions

Nevin Kapur Delabratra Ghosh Franc Brglez
CBL (Collaborative Benchmarking Laboratory)
Department of Computer Science
Box 7550, North Carolina State University, Raleigh, N.C. 27695, U.S.A.
http://www.cbl.ncsu.edu/demos

Abstract

Today, typical experiments that report 'performance' of EDA algorithms are based on isolated instances of circuit benchmarks. In this paper, we design and report results of experiments that use a large number of sample circuits from several equivalence classes. We consider a number of known circuit benchmarks and construct large samples of circuits that belong to the same equivalence class. Specifically, we examine properties of the recently introduced equivalence class of signature-invariant mutants $\eta_i \in N_{\sigma_0}(i > 0)$, where the signature $\sigma_0$ is based on a single reference circuit $\eta_0$.

The main goals of the experiments in this paper are: (1) to introduce a methodology that objectively measures the performance of algorithms used repeatedly in typical EDA tools, (2) to critically examine data sets that can be or should be used to support such a methodology. We argue that the proposed selection of mutants from the respective equivalence class represents an important set of benchmarks since we can consistently show that design objectives, being optimized by various algorithms, behave as random variables and give rise to well-defined distributions of circuit mutants.

I. Introduction

Scientific experiments are characterized by extensive measurements that can be independently repeated and statistically verified by others. In contrast, methods to measure the performance of EDA algorithms and tools are ad hoc and sparingly recorded. There are various reasons for this. To a large extent, planning and executing a series of benchmarking experiments is hampered by the lack of suitable test data. As circuits, and thereby problems to be investigated, become larger, it may well be impossible to create a large enough data sample to serve as a basis for statistically significant benchmarking experiments. Relative to other disciplines, the origins of EDA algorithms and tools go back only a few decades, while demands for their performance continue to increase unabated — there is no period of relative stability.

Random graphs, a potentially unlimited source of benchmarks, have not been accepted as realistic circuit benchmarks. However, new approaches to random circuit generation, that take into consideration constraints of digital circuits, have been reported recently [1], [2]. Both approaches have been motivated by the need to generate a large number of circuits to test FPGA architectures. In [1], the approach is guided by the objective of achieving a target Rent's exponent. In [2], circuit generation is driven by a profile based on a large number of realistic circuits. Both approaches demonstrate that random circuits, as generated by the respective algorithms, have characteristics that are useful for the target applications. Occasionally, both have difficulty in completing all targeted connections.

In this paper, we analyze the recently introduced equivalence class of signature-invariant digital circuits, mutants $\eta_i \in N_{\sigma_0}(i > 0)$, where the signature $\sigma_0$ is based on a single reference circuit $\eta_0$. Typically, the reference circuit $\eta_0$ represents a netlist of a known design. The signature-invariance guarantees that $\eta_0$ and all of its mutants $\eta_i$ are represented as netlists with the same number of I/Os, the same number of nodes, and the same number of pins distributed across the same number of levels.

Mutants $\eta_i$ as defined here, are not random netlists. Rather, experiments analyzed in this paper are based on an unbiased random selection of up to 100 mutants from the huge equivalence class $\eta_i \in N_{\sigma_0}(i > 0)$. We argue that each such selection of mutants from the equivalence class represents an important set of benchmarks since we can show that (1) design objectives being optimized by various algorithms behave as random variables, giving rise to well-defined distributions of circuit mutants, (2) the relative position and the shape of the distributions in the same equivalence class depends on the choice of the optimization algorithm.

The paper is organized as follows. Section 2 motivates the approach by summarizing an experiment with sorting algorithms, setting the tone for the benchmarking experiments in the rest of the paper. Section 3 provides a brief overview of equivalence classes and mutants to support the context of data we use in the experiments. The mutant synthesis process is quite involved and is described elsewhere [3]. Section 4 proposes an approach to designing a series of benchmarking experiments using the notion and availability of a large number of circuits from the same equivalence class. Section 5 reports results of extensive series of experiments, leaving room for additional experiments by others. Section 6 presents conclusions and a call for participation.

II. Motivation

To motivate and set the tone for the benchmarking experiments, we summarize an experiment that we carried out on two well-known and documented sorting algorithms: HeapSort and QuickSort.

We have implemented two programs in C under UNIX that execute on a SUN20 and SUN3 workstations: a version of HeapSort and a version of QuickSort. Both programs implement the respective algorithms described as pseudo-code in
Both algorithms are textbook cases with known worst-case and average performance behavior, thus the results of our experiment should be easy to explain. The insights gained from the experience with these algorithms could then be applied to other heuristic EDA algorithms whose behavior may be analyzed through experimental observations only, e.g., logic synthesis and optimization, technology mapping, partitioning, place and route, etc.

Can we measure time to execute a sort? A reasonable measure of performance of a sorting algorithm may be viewed as the time it takes to sort a set of numbers. However, as Figure 1 illustrates, this may not be the best strategy to employ. In the experiments in Figure 1, the same set of 4096 randomly generated numbers were repeatedly sorted on two machines. Notably, there is an inherent error in the measurement of time to sort as it is difficult to isolate the influences of factors like system load, read/write time, etc. Another point of interest is the fact that times can be radically different on two machines. For example, if one compares the HeapSort on Machine 1 versus the QuickSort on Machine 2, there seems to be a significant difference in their performance. However, QuickSort on Machine 1 and HeapSort on Machine 2 seem to be much closer in their performance. It is clear that a less ambiguous figure of merit is required.

Counting the number of comparisons. A figure of merit that is independent of the platform used in the experiment but can measure the performance of a sorting algorithm is the number of comparisons required to sort a given set of items. This figure is widely accepted as an accurate measure of the performance of sorting algorithms. It is independent of factors such as machine load, programming language, coding tricks, etc. Figure 2 shows the results of two kinds of experiments with QuickSort and HeapSort. In the first kind of experiments, 1024 unique random numbers between 0 and 1 were generated. The uniqueness condition was enforced to ensure that the data was not biased towards any particular algorithm. Following this, 100 sets of random permutations of 1024 unique numbers were used as inputs to the two sorting algorithms, keeping track of the number of key comparisons required to arrive at the sorted data. In the second kind of experiments, 100 sets of random numbers, each containing 1024 numbers, were used as inputs to the sorting algorithms, keeping track of the comparisons again.

**Observations.** As can be seen from Figure 2, a distribution is obtained for each of the experiment. For both sets of data, QuickSort outperforms HeapSort. The standard coefficient of variation (scv) for QuickSort is much higher than that for HeapSort, indicating that though QuickSort is faster on the average, its performance deviates from the average more than that of HeapSort. Both of these observations confirm what is already known from formal analysis of these algorithms.

Another interesting observation is that the results are virtually indistinguishable whether one starts with a single set of numbers and then permutes them randomly 100 times, or one takes 100 sets of random numbers. This can be attributed to the fact that the performance of both these algorithms depends entirely on the initial ordering of the keys.

Finally, Figure 3 summarizes the results obtained by performing experiments as described earlier for increasingly larger data sets – 32, 64, 128,..., 4096. Both QuickSort and HeapSort show a $O(N \log N)$ complexity. This is consistent with our theoretical knowledge about these two algorithms. Though there are cases when QuickSort can take $O(N^2)$ comparisons to arrive at an answer, on the average it performs better than HeapSort as the constants hidden in the $O()$ are smaller for QuickSort than for HeapSort.
Applications to proposed benchmarking experiments.

We have demonstrated that the objective to be optimized in the case of sorting algorithms – the number of key comparisons – does behave as a random variable, giving rise to a near-normal distribution of test cases. The mean and the variance of the distributions do not depend on the method by which the test cases have been generated: (1) via the random choice from the set of all permutations of unique numbers from a single list, or (2) by choosing all numbers in all tests cases randomly. To evaluate the performance of common EDA algorithms in a manner that is similar to the analysis presented in this section, we require netlists that are in the same equivalence class. Next section summarizes an approach to generation of such netlists. In Section V, we demonstrate their effectiveness.

III. EQUIVALENCE CLASSES, SIGNATURE, AND MUTANTS

Drawing an analogy with data sets we used to analyze the performance of sorting algorithms, we can consider two basic equivalence classes of netlists for any reference netlist: isomorphic and non-isomorphic class. The isomorphic class consists of netlists whose graphs are isomorphic and implement the same Boolean functions. Such classes can be generated readily by permuting the order of nodes in the netlist [5]. While useful, this class is not sufficiently general. We define a non-isomorphic class as one that preserves the ‘complexity’ of the reference netlist \( \eta_0 \) by maintaining the same number of I/Os, the same number of nodes, and the same number of pins distributed across the same number of levels. We capture properties of this class in a netlist signature \( \sigma_0 \) and define a class of signature-invariant digital circuits, mutants \( \eta_i \in N_{\sigma_0}(i > 0) \), where \( N_{\sigma_0} \) is the non-isomorphic equivalence class. The subject of equivalence class mutant synthesis itself is beyond the scope of this paper and is available in [3].

We use a simple example circuit from [6] to illustrate and motivate the introduction of the equivalence class of circuit mutants. In Figure 4, we represent all circuits as directed acyclic bipartite graphs in topological order: at each level \( i \) we have a set of logic nodes driving a set of fanout nodes. The set of fanout nodes at level \( i \) drives a set of logic nodes at level \( i + 1 \). To represent a circuit as a bipartite graph, we break and reconnect each net that spans more than one level through a feedthrough node, and draw it as a ‘buffer-like’ node. In Figure 4(a), we have 17 logic nodes assigned to levels 1 through 6: 14 nodes are denoted as 2-input combinational nodes representing the gates, 3 nodes are feedthrough nodes. For simplicity of notation, we distinguish at most 5 types of logic nodes: primary inputs \( L_I \), primary outputs \( L_O \), feedthroughs \( L_A \), inverters and buffers \( L_I \), and 2-input combinational nodes \( L_2 \). By assigning a level number \( i \) to each node, we can form five distributions. The set of all five such distributions forms a signature \( \sigma \) of a directed acyclic bipartite graph. The signature \( \sigma_0 \) in Figure 4(e) induces an equivalence class \( N_{\sigma_0} \) of mutants. By inspection, we find that circuits in Figure 4(c-d) belong to \( N_{\sigma_0} \) and so does the reference circuit in Figure 4(a).

We can prove that signature of the directed acyclic bipartite graph as defined here is unique. Also, we represent a synchronous sequential circuit with loops as an acyclic bipartite graph where loops are broken by minimal number of pseudo-primary inputs and outputs. The corresponding signature of the sequential circuit requires that we distinguish at least 12 types of logic nodes in its bipartite graph representation.

This particular example relates to simple experiments with a state-of-the-art logic synthesis tool. The circuit in Figure 4(a) has 14 combinational nodes at 6 levels, performs a simple arithmetic function, and serves as an input to a logic synthesizer SIS [7]. Surprisingly, the circuit resynthesized with a nominally powerful script boolean results in a larger circuit in 4(b). The question arises whether this is an isolated case or whether there exist circuits in the same equivalence class that could introduce even stronger challenge for the same script of SIS. The answer is yes: the two mutants in Figure 4(c-d) have 19 and 18 nodes, 7 and 9 levels, respectively.
the conspicuous exception of randomization [9]. It was Fisher who has shown how tests of significance and confidence limits can be constructed, using only the fact that randomization has been properly applied in the experiment.

The central purpose of the theories of experimental design, founded by Fisher, is to maximize, for a given experimental effort, the amount and the reliability of the information which can be obtained. At this point, the design of benchmarking experiments as proposed in this paper is relatively simple and does not rely on the vast body of literature and textbooks that provide the background in this field [10], [11], [12]. However, this may change. There are many opportunities to improve the benchmarking experiments as they relate to algorithms and tools in EDA.

At present, we can classify the benchmarking experiments in EDA as having one of the three modes:

- **Mode 1**: scoring a single test against a large sample of algorithms/tools in the same equivalence class;
- **Mode 2**: scoring a large sample of tests in the same equivalence class against a single algorithm/tool;
- **Mode 3**: scoring a large sample of tests in the same equivalence class against a large sample of algorithms/tools in the same equivalence class.

In a Mode 1 experiment, one can argue that the choice of the sample of tools has been unbiased and random for the chosen test if the outcome of the experiment gives rise to an approximately normal distribution, such as shown in Figure 5-a. Similarly, in a Mode 2 experiment, one can argue that the choice of the sample of tests has been unbiased and random for the chosen tool if the outcome of the experiment gives rise to an approximately normal distribution, such as shown in Figure 5-b. A Mode 3 experiment is a combination of Mode 1 and Mode 2. The major emphasis of this paper is on a set of experiments in Mode 2.

Instances of Mode 1 and Mode 2 experiments are well known. Figure 5-a may represent a scaled distribution of SAT or GRE scores for a large random sample of student applicants; the target shooting scores of a large random sample of competitors, etc. Figure 5-b may represent a scaled distribution of target shooting scores for a single competitor, given a large random sample of ammunition.

**Design of EDA Benchmarking Experiments.** As to benchmarking experiments of EDA, experiments in Mode 1 may not be practical; it is not possible to gather enough diverse algorithms/tools into a single environment for a controlled experiment. A case in point are the numerous listing of benchmarking experiments in the current literature: viz., many of them may report on results in logic synthesis, not all for the same set of benchmarks, and not all in terms of standardized metrics. Also, we feel that the results of a single comparative test cannot decide the pros and cons of an algorithm or a tool.

Up to now, the major problem with Mode 2 experiments has been that there are not sufficient test cases, i.e., benchmarks in the same class, to complete an experiment such as shown in Figure 5-b. We are aware of a single instance of reporting experimental results that would correspond to Mode 2. The paper in [5], reports distributions for a number of layout algorithms/tools, using the isomorphic equivalence circuit class of circuits (permitting the order of nodes in the netlist). The equivalence classes of mutants introduced in this paper have been designed to facilitate design of experiments such that data can be gathered according to the Mode 2.

We are considering two workflows in the design of experiments presented in this paper. Results, based on the concepts introduced in this workflow are presented in Section V and include schematics generation, netlist partitioning, technology-independent logic optimization, technology-specific logic optimization (technology mapping), and standard cell placement and routing. The latter is carried out to the point of mask layout of each circuit mutant whose layout area and channel wire length is measured explicitly. The workflow in Figure 6 considers equivalence classes of circuit mutants to analyze the behavior of algorithms/tools in Mode 2.

**Fig. 6.** Equivalent class mutant synthesis as applied to standalone Mode 2 benchmarking experiments.

The pairwise comparative workflow in Figure 7 illustrates the principle of comparing, according to Mode 3, two algorithms/tools driven by a common set of equivalent class mutant circuits. In addition, we also process all optimized circuit mutants using a common set of tools: e.g., technology mapping and layout generation. This way, we can measure whether the effects of optimization at an early stage of the design process will actually propagate and make a difference in the final design objective: in this case, circuit area at the mask level.

**Reference Circuits.** Our first target set of reference circuits to be included for mutant generation is listed in Table I. Most of these circuits have been characterized in the IWLS’91 User Guide [13]. We also include the simple circuit Roth’62 since the circuit and many of its mutants still represents a challenge to synthesis tools today. In addition, as a reference circuit in Figure 4, it also readily illustrates the basic concepts about the signature and its signature-invariant mutants.
CPU intensive: it includes logic synthesis, optimization and technology mapping, as well as standard cell layout for all 101 circuit mutants in each class. Our current experimental results raise a number of probing questions, thus the fact that we use a very uniform representation of all netlists can only accelerate finding some definitive answers.

All mutants that we have completed to date are available on the web [13]. As more runs are completed, we will post additional results and mutants in the same directory.

V. EXPERIMENTAL RESULTS

Planning, executing, and presenting the results of a large set of benchmarking experiments is a major challenge – more so if such experiments are to be independently reproduced and improved by others. When considering large classes of mutant circuits, processing and management of input and output data requires a high degree of automation. In the near future, we anticipate that much of the data generation, experiment execution, and comparative report generation of final results will not only be largely automated but distributed among several collaborating sites. An Internet-based workflow technology, such as discussed in [16], may facilitate this effort.

Results reported in this section introduces a snapshot of experiments currently completed. We expect them to catalyze an in-depth analysis of the intrinsic limitations of current algorithms, that are clearly challenged even by some of the simplest equivalent class mutant circuits reported below.

Mode 2 Experiment in Figure 8. As an example, consider the reference circuit $\eta_0 = C499$. The signature-invariance guarantees that $\eta_0$ and all of its mutants $\eta_i$ are represented as netlists with the same number of I/Os, the same number of nodes and the same number of pins distributed across the same number of levels. Upon submitting this set of mutants to (a) a schematic generator [17] which minimizes the wire crossings at each level, and (b) a bi-section partitioner [18] which minimizes the mincut, we observe that both the wire crossings and the mincut are random variables, giving rise to approximately normal distributions such as shown in Figure 8. This is not a coincidence, as reported by results of additional experiments in this section.

The largest new circuit $avq$ represents a recent CMOS design that has already been used in the Place and Route Contest (TimberWulf Hunt) held during the International Workshop on Layout Synthesis in 1992 [14]. Our goal is to introduce large benchmark equivalence classes that can be used by researchers not only in traditional placement and routing, but also in logic optimization, test generation, verification, BDD generation, timing-driven partitioning, etc. In this case, we require a simple directed hypergraph representation that preserves the signal direction and logical functionality of each node. The original $avq$ circuit representation is not suited for this task since the original netlist contains wired logic. Such descriptions cannot be routinely handled by typical logic optimizers, timing verifiers, and test generators, nor could we consider generating an unambiguous circuit signature and related equivalence classes of mutants. We removed all wired logic before proceeding to the final step of bringing all netlists to a common two-input node representation for signature characterization.

In this paper, we report experimental results for four equivalence classes of circuits in Table I: C499, C1355, X3, and C6288. Each class consists of 101 mutants: the first circuit is the reference circuit, all other circuits are randomly chosen mutants. The full characterization is time consuming and

![Fig. 7. Equivalent class mutant synthesis as applied to embedded Mode 3 benchmarking experiments.](image)

![TABLE I Reference Circuits for Mutant Equivalence Classes.](image)

![Fig. 8. Examples of circuit distributions based on the mutant equivalence class of C499: (a) produced by a schematic generator, (b) produced by a bi-section partitioner.](image)
out area and wiring just by permuting the order of nodes in the netlist in [5]. Our experiences with technology mapping as well as the layout tool available to us reveal only negligible variations compared to variations induced by signature-invariant mutants and reported in Figure 9, 10 later in this section.

However, for two particular algorithms, dot in [17] and PROPl in [18], the isomorphic equivalence circuit class itself can induce a distribution that is almost comparable to the one induced by the signature-invariant and shown in Figure 8. Specifically, we can observe up to 6% and 5% in variations in reported wire crossing and mincut (compared to 6.5% and 8% in Figure 8) just by re-executing the tools on 100 permuted netlists of C499!!

**Mode 2 Experiment in Figure 9.** Here, histograms profile the performance of the technology-specific optimization algorithm SIS-map. The cell library we used is libgenlib [13]. The primary objective of the algorithm is to minimize total area occupied by the cells. No delay optimization option was used. Since all mutants have the same number of 2-input nodes, one would expect only modest variations in the mapped results. However, we observe a significant amount of area and delay variation in each of the equivalence classes.

- Traditionally, considerable effort has been expended to minimize the ‘number of literals’, or equivalently, 2-input nodes, in the technology independent phase, with the expectation that even a modest reduction of the nodes will somewhat reduce the total area occupied by the cells. Mutants with the same number of 2-input nodes, when presented to SIS-map, induce a near-normal distribution with significant variance in total mapped area for each of the classes. The challenge for any other technology mapper is whether and how much it can reduce the mean and the variance of any of the mutant classes shown here. What other information is present in these classes of mutants that can significantly influence the outcome of the technology mapper?

Notably, except for mutants in class of C499, ALL mutants in other classes map to a larger area and more delay than the reference circuit – indicating that if we generate a larger number of mutants, there are only a few mutants as ‘easy’ for THIS layout generator as the reference circuit. However, as the mutant class of C499 shows, the situation can also be reversed!!

**Mode 3 Experiment in Figure 10.** Here, histograms profile the performance of a layout generator VPNR in OASIS [19]. We re-emphasize that in this case, the equivalence classes of mutants have the same number of 2-input nodes; moreover, all cells are of the same size in order to isolate the effect of interconnect on placement and routing. Again, we observe a significant amount of variation in the total layout area as well as the channel wire length as reported by OASIS. In the equivalence class of X3, layout area varies from 5.9 to 10.7 mm². In the equivalence class of C628, layout area varies from 8.6 to 291.7 mm².

- Traditionally, considerable effort has been expended to minimize the total area occupied by the cells before the layout generation phase. Mutants with the same cell area, when presented to VPNR/OASIS, induce a near-normal distribution with significant variance in total layout area for each of the classes. The challenge for any other layout generator is whether and how much it can reduce the mean and the variance of any of the mutant classes shown here. What other information is present in these classes of mutants that can so significantly influence the outcome of the layout generator?

**Performance Profile for Experiment in Figure 10.** Based on the results reported in Figure 10, we can profile the performance of the layout generator VPNR/OASIS in a manner that is analogous to the performance evaluation of the sorting algorithms in Figure 3. As shown in Figure 12(a), the ‘Total Cell Area’ increases linearly with the number of nodes, since each point on this curve is a measure of the number nodes scaled by the area occupied by each node, in this case a 2-input NAND gate. The ‘Average Layout Area’, however increases at a much higher rate, implying that as the size of the circuit increases, much of the area is required to route the interconnect. This hypothesis is confirmed by Figure 12(b) which shows the ‘Average Channel Wire Length’. The remarkable similarity in the slope of this curve to the ‘Average Layout Area’ curve is an affirmation of the close correlation of ‘Layout Area’ and ‘Channel Wire Length’.

All our experiments have been performed with a library of NAND gates implemented in MOSIS-ncmos (scalable_cmos) 2 micron technology, with 2 metal layers. Given the netlist of the mutants, it should not be difficult replicate these layout experiments with another set of tools, using the same library and generate performance curves similar to Figure 12. Comparison of the relative slopes of curves would be a true measure of the relative performances of the two standard cell place and route algorithms.

**Mode 3 Experiment in Figure 11.** Here, histograms profile pairwise comparisons of embedded multi-distribution benchmarking applications, introduced in Figure 7. We compare the respective distributions at the direct outputs of each algorithm, technology-independent optimization in this case. We also process all optimized circuit mutants by a common set of tools: technology mapping and layout generation. There are visible but minor variations in the performance of SIS-arg versus SIS-rug as measured by the median (med), the sample average (avg), the standard deviation (std), and the sample coefficient variance (scv). These variations decrease significantly after performing technology mapping and generating the layout. Thus, while for a single circuit, SIS-rug may appear to offer some advantage over SIS-arg, one can argue that on the average, performance of both is equivalent. If we look at the variance, there is less variation in the performance of SIS-arg than there is in SIS-rug.

- Mode 3 experiments are clearly more complex and more subtle to interpret than the Mode 2 experiments. By improving techniques used in Mode 2 experiments, we will also improve the clarity and statistical significance of Mode 3 experiments.

**VI. Conclusions and Call for Participation**

Benchmarking experiments analyzed in this paper are based on an unbiased random selection of 101 mutants from four equivalence classes. All mutants whose analysis that we have completed to date are available on the web [15]. Additional results and mutants will be posted.

By studying the properties of such equivalence classes and extending them further, the research community stands to gain deeper understanding about the average behavior of the algorithms and about ways to improve their performance.
Fig. 9. Single distributions based on the equivalence class mutants of C499, C1355, X3, C6288: (a) cell area and (b) circuit delay optimized by SIS-map.

Fig. 10. Single distributions based on the equivalence class mutants of C499, C1355, X3, C6288: (a) layout area and (b) wire length optimized by OASIS.
Fig. 11. Pairwise comparisons of distributions based on the equivalence class mutants of C1355: (a) nodes optimized by SIS-alg vs. nodes optimized by SIS-rug, (b) cell areas optimized by SIS-map, (c) layout areas optimized by OASIS.

For example, what are the key parameters in any of the equivalence class mutants that can so significantly influence the variations reported in the total cell area by the technology mapper, size of the mincut by the partitioner, total mask area produced by the layout generator, etc.? 

Call for Participation. Current sets of experiments are but a small initial step towards a new benchmarking paradigm in EDA, one that will build on the experience and the discipline that characterizes experimental design in sciences - comprehensive documentation of extensive measurements that can be independently repeated and statistically verified by others. Harnessing the distributed resources through the Internet, this goal may be realized and sustained through long-term efforts of collaborating researchers.

We are interested in discussing options with potential participants about testing, hosting, and facilitating benchmarking experiments within the context described in this and [16].

For more details, send e-mail to benchmarks@cbl.ncsu.edu with the following information in the body of the message:

subscribe demos

Acknowledgements. We appreciate the access to tools used in this research: SIS from the team at UC Berkeley, PROP from Roman Kunzar from U. of Ljubljana (Slovenia), and DOT from Lucent Technologies, Bell Labs Innovations.

References