Evaluation of Placer Suboptimality Via Zero-Change Netlist Transformations

Andrew B. Kahng
CSE and ECE Departments
University of CA, San Diego
La Jolla, CA, 92093
abk@ucsd.edu

Sherief Reda
CSE Department
University of CA, San Diego
La Jolla, CA, 92093
sreda@cs.ucsd.edu

ABSTRACT

In this paper we introduce the concept of *zero-change transformations* to quantify the suboptimality of existing placers. Given a netlist and its placement from a placer, we formally define a class of netlist transformations that produce different netlists from the given netlist but have the same Half-Perimeter Wire Length (HPWL). Furthermore, the optimal HPWL value of the new netlists is no less than that of the original netlist. By applying our transformations and re-executing the placer, we can interpret any deviation in HPWL as a lower bound to the deviation from the optimal HPWL value. Such deviation is a measure of suboptimality. Using these transformations, the suboptimality of several existing academic and industrial placers is studied on the IBM benchmarks. Our results show that current placers are suboptimal for zero-change transformations with deviations in HPWL by up to 32% on the IBM (version 1) benchmarks. The specific nature of our transformations also pinpoints possible directions for improvement in existing placers.

Categories and Subject Descriptors: B.7.2 [Design Aids]: Placement and routing.
General Terms: Algorithms, performance.
Keywords: Benchmarking, placer suboptimality, wirelength.

1. INTRODUCTION

Total HPWL minimization is the most traditional placement objective. This is no surprise given that HPWL is equivalent to the Steiner minimal tree (SMT) cost for two-pin and three-pin nets, and is well-correlated for multi-pin (≥ 4) nets [7]. Placers minimize HPWL heuristically by using, for example, min-cut partitioners [4, 20, 21], quadratic or analytical solvers [15, 9], or simulated annealing [19]. HPWL is also the typically reported metric when comparing results of different placers on various benchmarks [5, 8, 2].

Given a benchmark circuit and a placer, *placement benchmarking*, or *placer suboptimality evaluation*, is the problem of finding how close the placer's result is to the optimal result for the given benchmark. The placement problem or HPWL minimization is notoriously hard since (1) it is NP-hard [18], (2) it has no polynomial constant approximation algorithms [17], and (3) it has no approximation schemes [18][1]. Given these theoretical results, researchers must rely on heuristic methods to solve the problem. Lack of placement benchmarking can lead to frustration since there is no direct way to assess whether existing heuristics are sufficiently close to optimal for arbitrary instances.

In this paper, we propose a new direction in placement benchmarking by introducing the concept of *zero-change transformations* and devising a set of such transformations to quantify the suboptimality of existing placers. Given a netlist and its placement from a placer, zero-change transformations alter the given netlist while keeping its HPWL constant, resulting in *zero change* to its HPWL. More importantly, the optimal placement HPWL of the new netlist has a value no less than the original netlist's optimal HPWL. Thus, by executing the placer on the new netlist, we can use any deviation of the new HPWL from the original HPWL as a lower bound on the deviation from optimal results for the new netlist. Our empirical results show that existing placers fail to reproduce their original HPWL results, with large deviations. One positive outcome of this work is the ability to extract useful suboptimality information with any given arbitrary benchmark.

The organization of this paper is as follows. Section 2 briefly summarizes previous work on the placement benchmarking problem. Section 3 gives a number of preliminaries and definitions essential to understand this work. Section 4 formally introduces the concept of zero-change transformations, and gives a number of such transformations. Experimental results from the application of different transformations to various netlists are given in Section 5. Finally, Section 6 summarizes the implications of our work and gives directions for future work.

2. PREVIOUS WORK

Several papers [10, 5, 8, 2, 16] tackle the benchmarking problem. Hagen [10] *et al.* quantify the suboptimality of VLSI layout heuristics, e.g., placers and partitioners, by scaling VLSI instances and comparing the results of VLSI layout heuristics against a pre-calculated value from the unscaled instances. A recent paper by Chang *et al.* [5] uses an overlooked construction method by Hagen *et al.* [10] to

[1]Assuming $P \neq NP$.

ISPD'05, April 3–6, 2005, San Francisco, California, USA.

optimally construct a number of benchmarks with known optimal HPWL. We note that this construction method has much earlier roots in the placement literature [11]. Existing placers, e.g., Capo [4], Dragon [20], mPL [6], and FengShui [21] are benchmarked on the optimally constructed benchmarks, and it is concluded that there exists a significant gap between the placers' HPWL results and the optimal placement HPWL. The drawback of [5]'s results is that the benchmarks are unrealistic given that only local signals are considered. To overcome this drawback, Cong *et al.* [8] added global hyperedges and established placement upper bounds. Their experimental results show that the performance of available placers approaches the upper bounds as the percentage of global edges increases. It is unknown whether the calculated upper bounds are tight or loose.

The results of different placers on various benchmarks are given in [2], where HPWL, timing and routability results of different placers are tabulated. The results show that placers exhibit different efficiencies on different benchmark families. For example, mPL [6] outperforms all other placers on the PEKO [5] benchmarks, but Dragon [20] outperforms other placers on the IBM benchmarks. Placer efficiency in the presence of various netlist structures is studied by Liu and Marek-Sadowska [16]. Using existing benchmarks and placers, the effects of net degree distribution, net count, and Rent's exponent are tabulated and practical conclusions are given. Kahng and Mantik [12] study the mismatch between incremental optimizers, e.g., partitioners and ECO placers, and instance perturbations. In another effort, the stability of different runs of Capo on the same benchmark is studied [1], where tying a number of randomly selected cells to their regions is proposed to stabilize results from different Capo runs.

3. PRELIMINARIES

A circuit netlist is a hypergraph $H = (V, E)$, where V is a set of vertices representing the circuit cells, and E is the set of hyperedges representing the circuit wires. A hyperedge $e \in E$ is a set of vertices $e \in 2^V$, where $|e|$ gives the *cardinality* or *degree* of hyperedge e. The placement area is composed of a number of sites that cells can legally occupy. Each cell may occupy a number of sites depending on its width. A placement of a hypergraph is defined as follows.

Definition 1. A *placement* π of a given hypergraph $H(V, E)$ is a mapping $\pi : V \to Z^+$ assigning a placement site to every netlist cell such that no two cells overlap. If a cell occupies more than one site then the mapping gives the first occupied site.

The Half-Perimeter Wire Length (HPWL) of a hyperedge e in a given placement π is the length of half the perimeter of the smallest bounding box that includes all vertices of e[2]. Such HPWL of a hyperedge is denoted by $l(e, \pi)$. The total HPWL (or wirelength) is $L(H, \pi) = \sum_{e \in E} l(e, \pi)$. A placement π_* of a hypergraph H that has minimum total HPWL is called an *optimal placement*. The HPWL of an optimal placement is called *optimal wirelength or HPWL*[3].

[2]Without loss of generality, we assume throughout this paper that hyperedges/nets are connected to cells via pins at the center of the cells.

[3]There can be more than one optimal placement yielding the same optimal HPWL.

A *suboptimal placement* is a placement with a total HPWL larger than the optimal HPWL.

Definition 2. A netlist or a hypergraph *transformation* applied to an input hypergraph $H_1 = (V, E)$ produces a new hypergraph $H_2 = (V, E')$ with the same set of vertices as H_1 but with a different set of hyperedges, i.e., a netlist transformation changes the connectivity.

With these basic definitions, we are ready to introduce the concept of zero-change transformations.

4. ZERO-CHANGE TRANSFORMATIONS

In this section we introduce the concept of zero-change transformation as well as a number of such transformations.

Definition 3. Given a placement π_1 of some hypergraph H_1, a netlist transformation to H_1 produces a new hypergraph H_2 with the same number of vertices as H_1 but with a different set of hyperedges. We define this transformation as *zero-change* if the following two properties are satisfied:

- **Quiescency Property**: $L(H_1, \pi_1) = L(H_2, \pi_1)$, i.e., the transformation results in zero-change to HPWL with respect to the input placement π_1.
- **Hardness Property**: For any other placement π_k : $L(H_1, \pi_k) \leq L(H_2, \pi_k)$.

From the hardness property, it is possible to qualify the relationship between the optimal placement π_1^* of H_1 and the optimal placement π_2^* of H_2.

Theorem 1. Given an original hypergraph, a hypergraph generated from zero-change transformations has an optimal HPWL no less than that of the original hypergraph, i.e., $L(H_2, \pi_2^*) \geq L(H_1, \pi_1^*)$.

Proof. Towards a contradiction, assume that the optimal placement π_2^* of H_2 has HPWL less than the optimal placement π_1^* of H_1. Using π_2^* for H_1 gives a placement with HPWL $L(H_1, \pi_2^*) \leq L(H_2, \pi_2^*)$ from the hardness property. Consequently $L(H_1, \pi_2^*) < L(H_1, \pi_1^*)$, contradicting the assumption that π_1^* is the optimal placement of H_1. $\square$

The use of zero-change transformations for benchmarking is illustrated in Figure 1. Given a netlist H, placer P produces a placement π_1 with HPWL $w_1 = L(H_1, \pi_1)$. Given π_1, applying zero-change transformations to H_1 produces a new netlist H_2. From the quiescency property, $L(H_2, \pi_1) = L(H_1, \pi_1)$. However, executing P on H_2 might produce a new placement π_2 with some wirelength $w_2 = L(H_2, \pi_2)$. The main question is whether $w_1 = w_2$. If the placer is optimal then $w_1 = w_2$. If the placer is suboptimal then there are three possibilities:

1. $w_1 = w_2$ indicating that the placer is stable and not sensitive to the netlist transformations.
2. $w_2 < w_1$ indicating that the original placement was not optimal and the transformations lead the placer to a better suboptimal placement.
3. $w_2 > w_1$ showing that the placer is suboptimal and sensitive to the netlist transformations. Since w_1 acts as an upper bound to the optimal placement of H_2, the amount $w_2 - w_1 = L(H_2, \pi_2) - L(H_2, \pi_1)$ is a lower bound to the suboptimality gap of H_2 which is equal

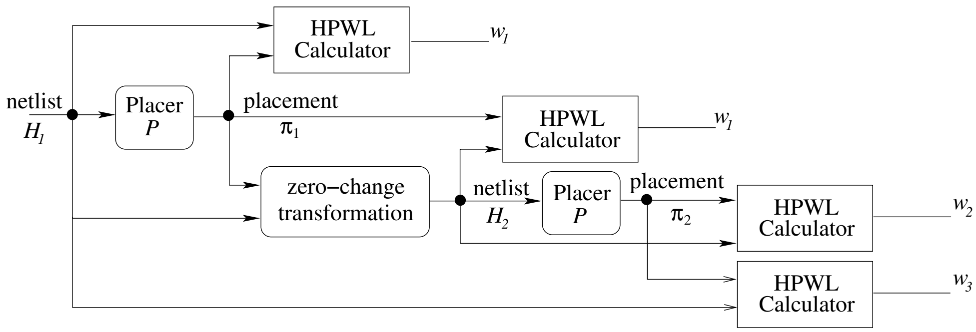

Figure 1: Conceptual presentation of zero-change transformations. The difference $w_2 - w_1$ represents a suboptimality measure of placer P.

to $w_2 - L(H_2, \pi_2^*) = L(H_2, \pi_2) - L(H_2, \pi_2^*)$, as shown in Figure 2.

The most important characteristic of zero-change transformations is that the optimal placement HPWL of the new netlist H_2 is no less than that of the original netlist H_1 as established in Theorem 1. Thus, executing the placer on the new netlist likely yields the third possibility where $w_2 > w_1$. This will be empirically demonstrated in Section 5.

One may wonder about $w_3 = L(H_1, \pi_2)$ produced from using π_2 for the original netlist H_1. This raises the possibility of using netlist transformations to improve the placeability of netlists. Notice that from the hardness property, we already know that $L(H_1, \pi_2) \leq L(H_2, \pi_2)$. We now propose a number of zero-change transformations to assess different placers' performance.

4.1 Hyperedge Cardinality Increase

The purpose of this transformation is to assess the sensitivity of placers to hyperedge cardinality by examining the impact of increasing the cardinality of hyperedges. We only increase the cardinality of hyperedges of degree ≥ 3. Our transformation is simple: given a netlist H and its placement π_1, the bounding box of each hyperedge (excluding 2-pin edges) is calculated, and an additional number of vertices are added to each hyperedge from within its bounding This cardinality increase procedure HYPERC is given in Figure 3. Before we prove that HYPERC is a zero-change transformation, we state the following lemma which is easy to prove.

Lemma 1. Given two sets of nodes S_1 and S_2: if $S_1 \subseteq S_2$ then $l(S_1, \pi_k) \leq l(S_2, \pi_k)$ in any placement π_k. □

Lemma 1 basically states that HPWL is *monotone* [3].

Theorem 2. Procedure HYPERC in Figure 3 is a zero-change transformation.

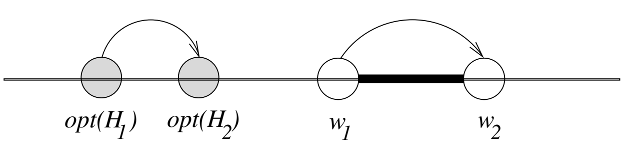

Figure 2: Relationship between the different HPWL quantities. $opt(H_1) = L(H_1, \pi_1^*)$ **is optimal HPWL of** H_1 **and** $opt(H_2) = L(H_2, \pi_2^*)$ **is the optimal HPWL of** H_2**.** $w_2 - w_1$ **is a lower bound on the suboptimality gap,** $w_2 - opt(H_2)$**, of the new netlist** H_2**.**

Proof: If the netlist produced by procedure HYPERC has the quiescency and hardness properties of Definition 3 then the theorem is proved. We will prove that each of these properties holds.

- Quiescency: Given a hypergraph H_1 and a placement π_1, applying procedure HYPERC produces a new hypergraph H_2. By construction, adding a number of vertices to a hyperedge from within its bounding box does not change its HPWL value. Therefore, $L(H_2, \pi_1) = L(H_1, \pi_1)$.
- Hardness: Given some $\pi_k \neq \pi_1$, H_1 would have an HPWL value of $L(H_1, \pi_k)$. Replacing each hyperedge e_i in H_1 with e_i' according to procedure HYPERC yields a HPWL value of $L(H_1, \pi_p) + \sum_i (l(e_i', \pi_k) - l(e_i, \pi_k)) \geq L(H_1, \pi_k)$ since $l(e_i', \pi_k) \geq l(e_i, \pi_k)$ by Lemma 1. Thus $L(H_2, \pi_k) \geq L(H_1, \pi_k)$. □

Finally, we note that the converse or "anti" transformation to HYPERC, where a hyperedge's cardinality is decreased, does not satisfy the zero-change requirements since it might yield a netlist with lower optimal HPWL than the original netlist.

4.2 Hyperedge Decomposition

Our second transformation simplifies a hyperedge by *decomposing* it into two hyperedges, with each of the new hyperedges having smaller cardinality than the original hyperedge. We define an optimal hyperedge decomposition as follows.

Definition 4. A hyperedge e is *optimally decomposable* in some placement π_k if it is possible to decompose e into two hyperedges e_1 and e_2 such that $l(e, \pi_k) = l(e_1, \pi_k) + l(e_2, \pi_k)$

Input: A hypergraph $H_1 = (V, E_1)$ and a placement π_1 for H.
Output: A new hypergraph $H_2 = (V, E_1)$.

1. Initialize $E_2 = E_1$.
2. For each hyperedge $e_i \in E_2$ where $|e_i| \geq 3$:
   3. Find the set of vertices C_{e_i} enclosed within the bounding box of e_i in placement π_1.
   4. If $C_{e_i} \neq \emptyset$ then augment hyperedge e_i as follows: $e_i = e_i \cup S_e$, where $S_{e_i} \subseteq C_{e_i}$.
5. Return hypergraph $H_2 = (V, E_2)$.

Figure 3: Procedure HYPERC for hyperedge cardinality increase.

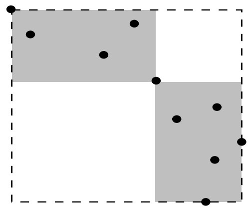

Figure 4: An example of an optimally decomposable hyperedge.

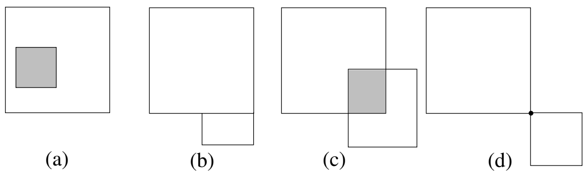

Figure 5: Enumeration of possible bounding box configurations.

and $e = e_1 \cup e_2$.

Figure 4 shows a hyperedge, whose bounding box is represented by a dashed line, optimally decomposed into two hyperedges as shown by the dashed rectangles. The following Lemma is crucial for our transformation.

Lemma 2. For any two hyperedges e_i and e_j with $|e_i \cap e_j| \neq \emptyset$, $\max(l(e_i, \pi_k), l(e_j, \pi_k)) \leq l(e_i \cup e_j, \pi_k) \leq l(e_i, \pi_k) + l(e_j, \pi_k)$ in any placement π_k.
Proof: The proof is by enumerating all possible cases for bounding boxes of e_i and e_j. Since $|e_i \cap e_j| \neq \emptyset$, there are only three possible configurations for the bounding boxes of e_i and e_j.

1. *Contained:* In this case, the bounding box of one hyperedge is completely contained within the other bounding box as shown in Figure 5.a. In this case $l(e_i \cup e_j, \pi_k) = \max(l(e_i, \pi_k), l(e_j, \pi_k))$.
2. *Overlapping:* In this case, the two bounding boxes overlap with $l(e_i \cup e_j, \pi_k) < l(e_i, \pi_k) + l(e_j, \pi_k)$ as shown in Figures 5.b and 5.c.
3. *Touching:* In this case, the two bounding boxes touch each other at a common vertex with $l(e_i \cup e_j, \pi_k) = l(e_i, \pi_k) + l(e_j, \pi_k)$ as shown in Figure 5.d. □

From the previous lemma, it is easy to see the following.

Lemma 3. e_1 and e_2 give an optimal decomposition of e in some placement π_k only if $|e_1 \cap e_2| = 1$ and the bounding boxes of e_1 and e_2 touch at their common vertex. □

Lemma 3 provides us with a simple characterization to optimally decompose any hyperedge. Using this characterization, we devise a a procedure for optimal hyperedge decomposition (procedure HYPERD) as given in Figure 6. We next prove that procedure HYPERD is a zero-transformation procedure.

Theorem 3. Procedure HYPERD is a zero-change transformation.
Proof: If the netlist produced by procedure HYPERD has the quiescency and hardness properties of Definition 3 then

Input: A hypergraph $H_1 = (V, E_1)$ and its placement permutation π_1.
Output: A hypergraph $H_2 = (V, E_2)$.

1. Iterate until there is no possible decomposition:
2. Set $E_2 = \emptyset$.
3. For each hyperedge e_i in E_1:
4. For each vertex $v_j \in e_i$:
5. If e_i can be partitioned into two sets e_i^1 and e_i^2 such that the bounding boxes of e_i^1 and e_i^2 touch each other at v_j
   then insert e_i^1 and e_i^2 into E_2 and goto Step 3.
   else insert e_i into E_2.
6. Set $E_1 = E_2$.
7. Return hypergraph $H_2 = (V, E_2)$

Figure 6: Procedure HYPERD for hyperedge decomposition.

the theorem is proved. We will prove that each of these properties holds.

- Quiescency: Since procedure HYPERD decomposes edges only optimally according to Lemma 3, it is clear that $L(H_1, \pi_1) = L(H_2, \pi_1)$.
- Hardness: Given some $\pi_k \neq \pi_1$, H_1 would have an HPWL value of $L(H_1, \pi_k)$. Replacing each hyperedge e_i in H_1 with e_i' according to procedure HYPERD yields a HPWL value of $L(H_1, \pi_k) + \sum_i (l(e_i^1, \pi_k) + (l(e_i^2, \pi_k) - l(e_i, \pi_k))) \geq L(H_1, \pi_k)$ since $l(e_i^1, \pi_k) + l(e_i^2, \pi_k) \geq l(e_i, \pi_k)$ by Lemma 2. Thus $L(H_2, \pi_k) \geq L(H_1, \pi_k)$. □

Before ending this subsection, we note that the converse or "anti" transformation to HYPERD, i.e., merging two touching hyperedges into one bigger hyperedge, does not satisfy the zero-change requirements since it might yield a netlist with lower optimal HPWL than the original netlist.

4.3 Edge Substitution

Our third transformation deals with edges, i.e., hyperedges of degree two. Our transformation does not change the cardinality of edges but rather increases the total number of two-pin edges. We start with the following fact that characterizes the triangle inequality in Manhattan or rectilinear metric.

Fact 1. If d_{ij} gives the Manhattan distance between two placement sites i and j then $d_{ij} \leq d_{ip} + d_{pj}$ for any site p, and if p lies within the bounding box defined by sites i and j then $d_{ij} = d_{ip} + d_{pj}$.

We leverage Fact 1 for netlist transformation as given in procedure EDGESUB of Figure 7, where we take every edge, calculate its bounding box, and substitute it with two edges by using a third vertex from within its bounding box. Such substitution can be carried more than once, effectively transforming an edge between sites i and j into a path between sites i and j as depicted in Figure 8.

Theorem 4. Procedure EDGESUB is a zero-change transformation.

Input: A hypergraph $H_1 = (V, E_1)$ and its placement π_1.
Output: A hypergraph $H_2 = (V, E_2)$.

1. Initialize $E_2 = E_1$.
2. Find a two-pin edge $\{u, v\}$ in E_2 and calculate its bounding box in π_1
3. Find a node p inside the bounding box of $\{u, v\}$
4. If such node p exists then
   5. Delete $\{u, v\}$ from E_2 and insert two new two-pin edges $\{u, p\}$ and $\{p, v\}$ in E_2.

Figure 7: Procedure EDGESUB for two-pin edge substitution.

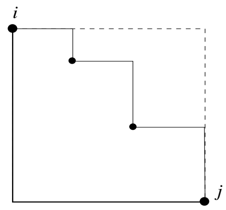

Figure 8: Substituting an edge between i and j by a path according to procedure EDGESUB does not change the optimality of a given placement.

Proof: We omit the proof for space limitations. Essentially, Fact 1 is used to prove Theorem 4 in a similar fashion to the proofs of Theorems 2 and 3. □

We note that if procedure HYPERC of Subsection 4.1 (for hyperedge cardinality increase) is allowed to operate on two-pin edges, then transformation EDGESUB can be considered as a combination of hyperedge cardinality increase (HYPERC) on two-pin nets, immediately followed by the hyperedge decomposition (HYPERD) of Subsection 4.2.

As a final remark, we stress that the converse or "anti" transformation to EDGESUB, by substituting a path of edges with a single edge, does not satisfy the zero-change requirements.

4.4 Hybrid Transformations

It is possible to apply the previous three transformations on a given netlist and empirically examine the collective impact of all transformations. Since each transformation is zero-change, the hybrid application of all transformations is also zero-change. We try the following combination:

1. Apply procedure HYPERD to decompose hyperedges into smaller hyperedges if possible.
2. Apply procedure HYPERC to increase the cardinality of hyperedges of degree ≥ 3 by four additional vertices if possible.
3. Apply procedure EDGESUB to increase the number of two-pin edges, substituting each edge by a path up to length four if possible.

5. EXPERIMENTAL RESULTS

In this section we empirically evaluate the suboptimality of existing placers with respect to our proposed transformations. This evaluation is carried out using the IBM benchmarks (version 1), four academic placers (Dragon [20] (version 3.01), Capo [4] (version 8.8 with feedback [14]), FengShui [21] (version 2.6), and mPL (version 4.0) [6]) and one industrial placer (Cadence's QPlace (version 5.2)). FengShui, Capo, and Dragon are based on a min-cut partitioning framework. mPL is a based a non-linear programming formulation in a multi-level optimization framework. Cadence's QPlace is believed to be based on a quadratic solver in a top-down partitioning framework. Since all pins are placed at the center of their respective cells in all circuits of the IBM benchmarks, we measure HPWL center-to-center and any additional pins necessitated by our transformations are also placed at the cells' centers. Before carrying out our experiments, we estimate the noise [1, 13] of different placers by reporting the average difference in HPWL for two different executions of a single placer on the same netlist[4]. Our results show that FengShui has a noise margin of around 0.92%, mPL has a noise margin of around 0.89%, Capo has a noise margin of around 2.9%, and Dragon has a noise margin of around 3.37%.

Our experimental execution flow is based on the outline of Figure 1. In all experiments, we report the percentage deviations SUBOPT= $\frac{w_2 - w_1}{w_1}$ and DELTA= $\frac{w_3 - w_1}{w_1}$. The first amount, SUBOPT= $\frac{w_2 - w_1}{w_1}$, is a lower bound on deviation from the optimal HPWL of the new netlists and thus is a measure of suboptimality for current placers. The second amount, DELTA= $\frac{w_3 - w_1}{w_1}$, is reported to see if the transformations can lead to an improvement in the placeability of the original netlists.

In a first series of experiments, we empirically determine the suboptimality of placers with respect to hyperedge cardinality increase transformations as given by procedure HYPERC. Table 1 gives the results of applying procedure HYPERC with cardinality increase of two when possible. The total number of pins (total hyperedge cardinality) in the original netlist **Pins** and in the new netlists after the transformation **New Pins** are reported. From the results, we observe that none of the five placers managed to maintain their original HPWL. All placers exhibit a substantial experience increase in HPWL.

To further study the impact of procedure HYPERC, we focus on the IBM01 benchmark and measure the HPWL changes due to a cardinality increase from 1 to a cardinality increase of 6. We plot our results in Figure 9. From the figure, current placers exhibit unstable and suboptimal behavior. Dragon shows a deviation of about 16% percent in HPWL. Capo shows a deviation by 24% percent in HPWL. QPlace shows a deviation by up to 18%. mPL and FengShui are relatively the least sensitive with deviations of up to 6% and 9% respectively.

In a second series of experiment, we empirically determine the suboptimality of placers to hyperedge decomposition as given by the HYPERD transformation. The empirical results are given in Table 2. Column **Nets** gives the original total number of nets for each benchmark, while column **New Nets** gives the total number of nets for each benchmark after applying procedure HYPERD. We report the percentage increase in each placer HPWL. From the results, we notice that placers sometimes are able to exploit such transformation to improve their results. On the average, there is an instability in the placers' results with a small average deviation in HPWL results. On the average, mPL deviates by 0.61%, Capo by 0.87%, FengShui by 1.32%, Dragon by

[4]Different placer's runs use the same netlist but with different ordering of nets.

bench	Pins	Placer	New Pins	SUBOPT	DELTA	bench	Pins	Placer	New Pins	SUBOPT	DELTA
ibm01	44266	Capo	50219	15.03%	9.08%	ibm05	126308	Capo	148739	5.30%	2.96%
		FengShui	50339	5.58%	2.87%			FengShui	148429	0.10%	0.10%
		Dragon	49627	11.96%	7.74%			Dragon	148494	1.14%	0.40%
		mPL	50336	9.33%	5.69%			mPL	149247	1.66%	0.77%
		QPlace	50057	14.20%	9.37%			QPlace	148117	1.10%	1.98%
ibm02	78171	Capo	92101	8.17%	5.03%	ibm06	124299	Capo	141852	2.29%	1.14%
		FengShui	91686	7.83%	4.49%			FengShui	142763	1.50%	0.59%
		Dragon	91029	8.47%	3.72%			Dragon	141855	11.39%	6.80%
		mPL	92238	6.51%	3.83%			mPL	142181	7.32%	4.58%
		QPlace	91472	4.83%	5.59%			QPlace	141121	4.33%	7.05%
ibm03	75710	Capo	87008	6.49%	3.98%	ibm07	164369	Capo	190746	9.42%	5.53%
		FengShui	86292	2.22%	1.39%			FengShui	189428	3.25%	1.55%
		Dragon	86505	3.16%	1.68%			Dragon	191139	11.10%	6.52%
		mPL	87653	1.72%	-0.53%			mPL	191315	2.78%	1.52%
		QPlace	87305	5.79%	6.87%			QPlace	191611	19.73%	17.47%
ibm04	89591	Capo	104903	2.85%	1.53%	ibm08	198180	Capo	225126	3.72%	1.85%
		FengShui	104855	3.53%	2.35%			FengShui	223595	2.04%	1.14%
		Dragon	105069	12.52%	8.57%			Dragon	224198	3.10%	1.80%
		mPL	106031	15.28%	9.68%			mPL	226102	2.08%	0.89%
		QPlace	104688	5.66%	6.96%			QPlace	225029	6.86%	6.67%

Table 1: Hyperedge Cardinality Increase Results. Cardinality Increased by Two. Pins is the total hyperedge cardinality in the original netlists. Pins New is the total hyperedge cardinality in the new transformed netlists. Average suboptimality (SUBOPT) results are as follows: FengShui 3.26%, mPL 5.84%, Capo 6.66%, QPlace 7.81%, and Dragon 7.85%.

bench	Nets	Placer	New Nets	SUBOPT	DELTA	bench	Nets	Placer	New Nets	SUBOPT	DELTA
ibm01	11507	Capo	12720	3.43%	2.53%	ibm05	28446	Capo	31462	2.93%	2.36%
		FengShui	12660	1.54%	0.90%			FengShui	31016	0.48%	0.17%
		Dragon	12632	3.92%	3.16%			Dragon	31463	-0.05%	-0.53%
		mPL	12848	2.16%	1.32%			mPL	31581	0.10%	-0.18%
		QPlace	12751	2.79%	1.79%			QPlace	31350	0.16%	-0.62%
ibm02	18429	Capo	20761	1.21%	0.51%	ibm06	33354	Capo	36711	-3.79%	-4.91%
		FengShui	20563	3.75%	3.01%			FengShui	36464	0.12%	-0.39%
		Dragon	20459	-0.21%	-0.97%			Dragon	36726	1.06%	0.45%
		mPL	20639	2.22%	1.74%			mPL	37084	-2.25%	-2.96%
		QPlace	20621	4.35%	3.60%			QPlace	36734	6.20%	4.77%
ibm03	21621	Capo	23673	1.22%	0.57%	ibm07	44394	Capo	49412	2.03%	1.21%
		FengShui	23490	1.95%	1.48%			FengShui	49056	3.17%	2.50%
		Dragon	23434	1.62%	1.21%			Dragon	49444	1.95%	-1.30%
		mPL	23980	0.00%	-0.66%			mPL	49684	-0.54%	-1.14%
		QPlace	23763	-2.57%	-3.39%			QPlace	49485	2.59%	1.64%
ibm04	26163	Capo	28887	1.11%	0.54%	ibm08	47944	Capo	53208	-1.23%	-1.95%
		FengShui	28711	2.47%	2.03%			FengShui	53122	-2.93%	-3.62%
		Dragon	28949	3.62%	2.84%			Dragon	53460	0.45%	-0.07%
		mPL	29294	-0.17%	-0.78%			mPL	54038	3.33%	2.74%
		QPlace	28987	-0.53%	-1.21%			QPlace	53111	2.26%	1.59%

Table 2: Hyperedge Decomposition Results. Nets is the total number of hyperedges in the original netlists. Nets New is the total number of hyperedges in the new transformed netlists. Average suboptimality (SUBOPT) results are as follows: mPL 0.61%, Capo 0.87%, FengShui 1.32%, Dragon 1.54%, and QPlace 1.90%.

1.54%, and QPlace by 1.90%. We also carry out a more detailed study on the ibm01 benchmark as given in Figure 10. In the plot, the x-axis gives the decomposition iteration, and the y-axis gives the HPWL produced from the different placers. While the magnitude of changes in HPWL is relatively small, there is a clear resemblance between the curves of Capo, QPlace, mPL, and FengShui. Dragon, however, seems to exploit the transformation to improve its performance. We can envision using this transformation within a placement run to simplify netlists, leading to better placements.

In a third series of experiments, we determine the suboptimality of existing placers with respect to the increase of two-pin edges by using transformation EDGESUB. We present our results in Table 3, where each edge is substituted by a path of length 2 when possible. We report the total number of hyperedges before the transformation (**Nets**) and after the transformation (**New Nets**). From the results, placers exhibit an unnecessary increase in wirelength by up to 20%. To further study the impact of procedure EDGESUB, we apply it to the ibm01 benchmark to transform its edges into paths of length l=2, 3, 4, and 5 and plot the results in Figure 11. It is important to stress that given one placement π_1, we apply procedure EDGESUB only once to substitute all edges by paths of length l if possible, i.e., we do not iterate executing procedure EDGESUB l times on consecutive placement runs. From the plot, we notice that placers exhibit a consistent increase in HPWL as the substitution increases with the HPWL deviating by up to 18% for academic placers and up to 35% for QPlace.

In a fourth series of experiments, we test the suboptimality of placers with respect to hybrid transformations. We give our results in Table 4. The results show that placers can deviate from their previous results by up to 32%.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we introduced a new concept *zero-change netlist transformations* to determine the suboptimality of existing placers. While one may envision many transformations that do not change the HPWL of a given netlist, our transformations share an important property: the optimal HPWL of new netlist is not less than the original HPWL

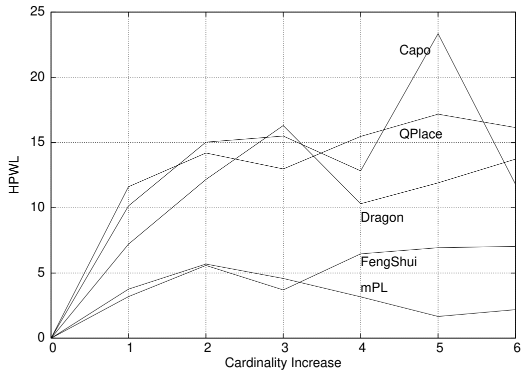

Figure 9: Effect of increasing the cardinality of hyperedges on the HPWL for the ibm01 benchmark.

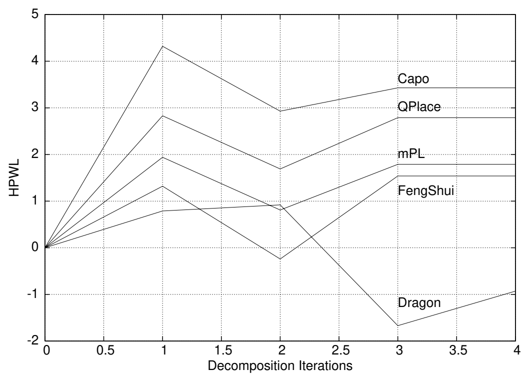

Figure 10: Effect of hyperedge decompositions on HPWL for the ibm01 benchmark.

optimal value, and consequently the placement of the new benchmarks is not "easier" than for the original benchmarks. By applying our transformations and re-executing a placer, we can interpret any deviation in HPWL results as a lower bound to the deviation from optimal HPWL value, and thus such deviation is a measure of suboptimality. Our set of netlist transformations can be summarized as follows.

- Procedure HYPERC increases, if possible, the cardinality of hyperedges with degree $\geq$ 3 leaving 2-pin edges intact.
- Procedure HYPERD simplifies large hyperedges (when possible) of degree $\geq$ 3 by decomposing a larger hyperedge into two or more smaller hyperedges.
- Procedure EDGESUB increases the number of two-pin edges if possible.

Our empirical results show that even when testing very few netlist variants, we can easily find large deviations in placement HPWL of up to 32%. From our empirical results, we make the following remarks.

- Remark 1: Placers' poor handling of hyperedges might be the main cause for HPWL increase due to procedure HYPERC transformations. This calls for more work on mechanisms such as terminal propagation, or hyperedge to clique/tree conversion. Perhaps better handling of such mechanisms can lead to substantial improvement.
- Remark 2: Empirical results indicate that placers exhibit large amount of suboptimality with respect to edge substitution. Such suboptimality might be a result of the typical top-down sequential placement process, where increasing the number of edges exemplifies the adverse impact of sequential optimization.

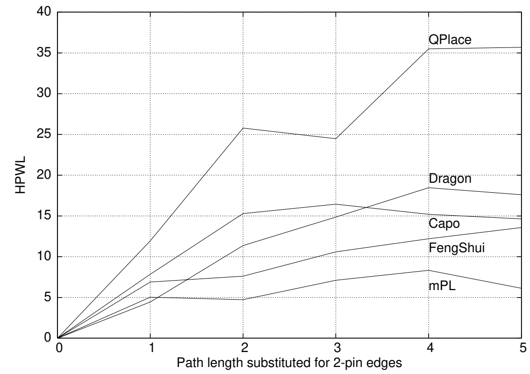

Figure 11: Effect of increasing the amount of edge substitution on the HPWL for the ibm01 benchmark.

- Remark 3: Placers can exploit the reduction in hyperedge cardinality by procedure HYPERD to improve their results. However, our procedure HYPERD simultaneously increases the number of hyperedges while reducing their cardinality. Thus, transformation HYPERD has the potential to reduce the HPWL (from Remark 1), and to increase the HPWL (from Remark 2).

Experimental researchers in physical design would no doubt agree there is a tendency to tune algorithms and codes to specific benchmarks [2]. A good placer should be good not just for a single real instance, but also for "similar" instances. Using our transformations allows the creation of a range of instances around any given arbitrary benchmark. Thus, for the first time, the field is afforded a means of creating "similar" instances in a systematic way such that the suboptimality and consistency of placement quality can be immediately evaluated

Our future work will focus on extending this work to (1) δ-change transformations, where the introduced transformations cause a δ change in the HPWL, but at the same time, the change in the optimal HPWL can be bounded, and (2) suboptimality evaluation using other metrics such as rectilinear minimum spanning trees, or Steiner trees.

7. REFERENCES

[1] S. Adya, I. Markov, and P. Villarrubia, "On Whitespace and Stability in Mixed-Size Placement," in *Proc. IEEE International Conference on Computer Aided Design*, 2003, pp. 311–318.

[2] S. N. Adya, M. C. Yildiz, I. L. Markov, P. G. Villarrubia, P. N. Parakh, and P. H. Madden, "Benchmarking for Large-Scale Placement and Beyond," in *Proc. ACM/IEEE International Symposium on Physical Design*, 2003, pp. 95–103.

[3] J. Beardwood, J. Halton, and J. Hammersley, "The Shortest Path through Many Points," *Proc. Cambridge Philos. Soc. 55*, pp. 299–327, 1959.

[4] A. E. Caldwell, A. B. Kahng, and I. L. Markov, "Can Recursive Bisection Alone Produce Routable Placements?" in *Proc. ACM/IEEE Design Automation Conference*, 2000, pp. 477–482.

[5] C. Chang, J. Cong, and M. Xie, "Optimality and Scalability Study of Existing Placement Algorithms," in *Proc. IEEE Asia and South Pacific Design Automation Conference*, 2003, pp. 621–627.

bench	Nets	Placer	New Nets	SUBOPT	DELTA	bench	Nets	Placer	New Nets	SUBOPT	DELTA
ibm01	11507	Capo	13703	7.84%	6.29%	ibm05	28446	Capo	35682	5.33%	4.27%
		FengShui	13852	6.89%	5.56%			FengShui	36173	1.50%	1.15%
		Dragon	13913	10.09%	8.48%			Dragon	36021	1.17%	0.78%
		mPL	13840	6.44%	5.03%			mPL	36399	1.97%	1.58%
		QPlace	12736	11.94%	9.37%			QPlace	32017	2.93%	1.98%
ibm02	18429	Capo	22461	4.28%	3.53%	ibm06	33354	Capo	42726	4.54%	3.75%
		FengShui	22922	6.03%	5.09%			FengShui	43096	4.15%	3.08%
		Dragon	22832	8.90%	7.35%			Dragon	43433	6.97%	5.56%
		mPL	22881	3.43%	2.90%			mPL	43551	10.13%	7.88%
		QPlace	21072	7.58%	5.59%			QPlace	36922	9.81%	7.05%
ibm03	21621	Capo	27217	6.76%	5.61%	ibm07	44394	Capo	54858	7.11%	5.70%
		FengShui	27617	4.16%	3.41%			FengShui	55755	3.20%	2.52%
		Dragon	27860	7.97%	5.86%			Dragon	56329	12.52%	10.60%
		mPL	27856	1.88%	0.93%			mPL	56768	4.90%	3.76%
		QPlace	23928	9.26%	6.87%			QPlace	49916	20.08%	17.47%
ibm04	26163	Capo	33153	5.06%	4.07%	ibm08	47944	Capo	58944	6.10%	3.71%
		FengShui	33373	3.97%	3.13%			FengShui	59811	3.89%	3.21%
		Dragon	34044	11.05%	9.61%			Dragon	60360	8.59%	7.23%
		mPL	34204	12.19%	10.69%			mPL	60528	8.37%	7.14%
		QPlace	29320	9.20%	6.96%			QPlace	53672	8.83%	6.67%

Table 3: Edge substitution Results. Nets is the total number of edges in the original netlists. Nets New is the total number of edges in the new transformed netlists. Average suboptimality (SUBOPT) results are as follows: FengShui 4.67%, mPL 5.84%, Capo 6.17%, Dragon 8.88%, and QPlace 10.78%.

bench	Nets	Placer	New Nets	SUBOPT	DELTA	bench	Nets	Placer	New Nets	SUBOPT	DELTA
ibm01	11507	Capo	20667	25.52%	15.74%	ibm05	28446	Capo	58780	10.13%	6.01%
		FengShui	20782	11.07%	5.97%			FengShui	62294	4.42%	2.15%
		Dragon	22010	19.11%	11.86%			Dragon	59834	4.51%	2.73%
		mPL	25503	13.39%	7.77%			mPL	95367	10.43%	5.94%
		QPlace	25724	24.45%	21.74%			QPlace	65487	3.96%	3.39%
ibm02	18429	Capo	38071	6.01%	3.07%	ibm06	33354	Capo	74137	6.41%	3.44%
		FengShui	38222	12.39%	7.21%			FengShui	73185	8.20%	3.98%
		Dragon	38888	17.99%	11.55%			Dragon	76948	32.66%	22.93%
		mPL	50006	10.03%	6.00%			mPL	99769	13.91%	6.50%
		QPlace	46934	7.76%	6.68%			QPlace	89587	13.18%	11.52%
ibm03	21621	Capo	45256	24.71%	14.25%	ibm07	44394	Capo	91524	7.72%	4.12%
		FengShui	48530	9.83%	4.94%			FengShui	99200	16.09%	9.20%
		Dragon	49959	20.06%	12.36%			Dragon	100220	10.96%	6.06%
		mPL	65744	17.32%	9.26%			mPL	139429	20.01%	10.27%
		QPlace	55863	12.51%	11.02%			QPlace	111255	9.50%	8.29%
ibm04	26163	Capo	56864	7.71%	3.86%	ibm08	47944	Capo	96031	7.58%	4.11%
		FengShui	55102	6.80%	3.43%			FengShui	101847	16.67%	7.42%
		Dragon	59298	10.94%	6.26%			Dragon	105234	10.61%	6.16%
		mPL	84333	37.29%	25.46%			mPL	134085	16.81%	9.95%
		QPlace	68422	12.94%	11.23%			QPlace	115639	15.94%	13.93%

Table 4: Hybrid transformation Results. Nets is the total number of edges in the original netlists. Nets New is the total number of edges in the new transformed netlists. Average suboptimality (SUBOPT) results for are as follows: FengShui 10.68%, Capo 11.97%, Qplace 12.53%, Dragon 16.86%, and mPL 17.40%.

[6] C.-C. Chang, J. Cong, D. Pan, and X. Yuan, "Multilevel Global Placement with Congestion Control," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22(4), pp. 395–409, 2003.

[7] C. E. Cheng, "RISA: Accurate and Efficient Placement Routability Modeling," in *Proc. IEEE International Conference on Computer Aided Design*, 1994, pp. 690–695.

[8] J. Cong, M. Romesis, and M. Xie, "Optimality and Scalability Study of Partitioning and Placement Algorithms," in *Proc. ACM/IEEE International Symposium on Physical Design*, 2003, pp. 88–94.

[9] H. Eisenmann and F. M. Johannes, "Generic Global Placement and Floorplanning," in *Proc. ACM/IEEE Design Automation Conference*, 1998, pp. 269–274.

[10] L. W. Hagen, D. J. H. Huang, and A. B. Kahng, "Quantified Suboptimality of VLSI Layout Heuristics," in *Proc. ACM/IEEE Design Automation Conference*, 1995, pp. 216–221.

[11] M. Hanan and J. M. Kurtzberg, "Placement Techniques," in *In Design Automation of Digital Systems, M. A. Breuer Ed.*, 1972, pp. 213–282.

[12] A. B. Kahng and S. Mantik, "On Mistmatches between Incremental Optimizers and Instance Perturbations in Physical Design Tools," in *ICCAD*, 2000, pp. 17–22.

[13] ——, "Measurement of Inherent Noise in EDA Tools," in *International Symposium on Quality in Electronic Design*, 2002, pp. 206–211.

[14] A. B. Kahng and S. Reda, "Placemet Feedback: A Concept and Method for Better Min-Cut Placement," in *Proc. ACM/IEEE Design Automation Conference*, 2004, pp. 357–362.

[15] J. M. Kleinhans, G. Sigl, F. M. Johannes, and K. J. Antreich, "GORDIAN: VLSI Placement by Quadratic Programming and Slicing Optimization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 10(3), pp. 356–365, 1991.

[16] Q. Liu and M. Marek-Sadowska, "A Study of Netlist Structure and Placement Efficiency," in *ISPD*, 2004, pp. 198–203.

[17] M. Queyranne, "Performance Ratio of Polynomial Heuristics for Triangle Inequality Quadratic Assignment Problem," *Operations Research Letters*, vol. 4, p. 1986, 231-342.

[18] S. Sahni and T. Gonzalez, "P-Complete approximation problems," *Journal of the ACM*, vol. 23, pp. 555–565, 1976.

[19] W.-J. Sun and C. Sechen, "Efficient and Effective Placement for Very Large Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 14(5), pp. 349–359, 1995.

[20] M. Wang, X. Yang, and M. Sarrafzadeh, "DRAGON2000: Standard-Cell Placement Tool for Large Industry Circuits," in *Proc. IEEE International Conference on Computer Aided Design*, 2001, pp. 260–263.

[21] M. Yildiz and P. Madden, "Global Objectives for Standard-Cell Placement," in *Proc. IEEE Great Lakes Symposium on VLSI*, 2001, pp. 68–72.