Locality and Utilization in Placement Suboptimality *

UCLA Computer Science Dept. Technical Report

Jason Cong,[1] Michalis Romesis[2], Joseph R. Shinnerl[3], Kenton Sze[4], and Min Xie[1]
[1]UCLA Computer Science [2]Magma Design Automation, Inc. [3]Tabula, Inc. [4]UCLA Mathematics
{cong,xie}@cs.ucla.edu michalis@magma-da.com jshinnerl@tabula.com nksze@math.ucla.edu

June 27, 2006

Abstract

Recent suboptimality studies of leading placement tools have drawn attention to the limitations of currently available benchmarking techniques. Available synthetic circuits with known optimal placements lack both mixed size objects and non-local nets, and their optimal placements contain zero white space. In order to identify and quantify deficiencies remaining in existing placement algorithms, two new classes of benchmarking circuits with known optimal or provably near-optimal placements are introduced. The first has non-local nets and is derived by netlist transformation from a precomputed placement of a given circuit. The second supports a user-specified amount of uniformly randomly distributed white space and is derived from a precomputed placement of the macros in the netlist. An adaptation of this second set of benchmarks is used to estimate separately the suboptimality of global placement and that of legalization and detailed placement. Experiments on the new benchmarks highlight deficiencies in leading academic tools and indicate promising new directions for placement research.

1 Introduction

Placement is a critical step in VLSI design. Interconnect delay dominates system performance, and placement determines the interconnect more than any other step in physical design. The complexity of modern designs, however, makes estimation of suboptimality difficult [12, 19, 13]. Studies on simplified, synthetic benchmarks with known optimal-wirelength placements (PEKO [6]) initially suggested that many leading tools may produce solutions with excess wirelength from 60% up to 150% or more. These results have generated wide interest in both industry [11] and academia [15, 17, 19]. Recent progress in placement [4, 5, 14, 1] has reduced the wirelength gap on PEKO to about 12%–40%.

The PEKO benchmarks, however, have well-known limitations. Although their cell counts, net counts, and net-degree statistics match corresponding quantities in standard industrial benchmarks [2], the PEKO circuits are simplified in three key ways, in order to guarantee known optimal solutions. First, all cells are squares of the same size. Second, the known optimal placements for the PEKO circuits are packed layouts with zero white space. Third, all nets in an optimal PEKO placement are local — the netlist of a PEKO circuit is defined over cells arranged in a regular array, with adjacent cells grouped into local nets of minimum half-perimeter wirelength (HPWL).

Subsequent studies [8, 13] derived useful lower bounds on the HPWL suboptimality of placements of circuits with more realistic netlists. However, it is not known how close the bounds are to the the true suboptimality gaps. The PEKU circuits [8] add non-local nets to packed, uniform-grid PEKO layouts but sacrifice any assurance of optimality. Zero-change netlist transformations [13] preserve both module shapes and core utilization, but they quantify the sensitivity of a placement tool to netlist changes, not the suboptimality of a given placement on a given netlist.

The benchmarks described in this paper directly address several of the shortcomings in existing suboptimality benchmarks. Two new sets of placement examples are constructed, one targeting the role of non-local nets in suboptimality, and another targeting the role of white space and large variations in module sizes. The first set, MC, is a set of standard-cell circuits with non-local nets in known optimal placements. A given netlist is modified so as to render a given placement for it optimal for the new netlist. Cell dimensions and locations are not changed, net-degree statistics are matched exactly, and over 60% of the original netlist is left unchanged. The second set, PWS, incorporates a parametrized percentage of white space into a mixed-size placement which precisely matches given macro dimensions and locations as well as the net-degree distributions of the ISPD 2005 benchmark suite [16]. HPWL for the placements generated for the PWS circuits are proven to be less than 3% above optimal for most cases and within 8% of optimal on all cases.

Typically, mixed size placement proceeds in three stages: global placement, legalization, and detailed placement. In practice, global placement is terminated when iterations are observed to make little or no reduction in the objective and the module-area distribution is sufficiently uniform. How much of the optimality gap left by contemporary methods should be attributed to deficiencies in global-placement algorithms, and how much to legalization and detailed placement? On a real circuit, there is no way of knowing how far a cell is from its nearest optimal location, at any stage. On circuits with known optimal or near-optimal placements, however, it is possible to evaluate precisely the quality of any of the three engines in isolation from its counterparts. Thus, the benchmark circuits described here provide a more precise means of quantifying the relative effectiveness of the methods used in the three stages. Results estimating the separate suboptimality contributions of global placement and legalization and detailed placement are described in Section 4.

*Partial support for this work has been provided by Semiconductor Research Consortium Contract 2003-TJ-1091 and National Science Foundation Contracts CCF 0430077 and CCF-0528583.

2 MC Benchmark Construction

Each MC example has an optimal-wirelength placement in which over 50% of the nets are non-local. Module shapes, core utilization, and net-degree statistics match corresponding quantities in a given benchmark exactly. The MC construction is described in this section.

2.1 Monotone Chains

The definition of a monotone chain in a netlist uses simple ideas from both graphs and hypergraphs. First, consider a path P in a *graph* G whose vertices lie in the plane. Let P consist of n consecutive edges (e_1,..., e_n) connecting $n+1$ vertices (v_0,...,v_n), vertex v_i with coordinates (x_i, y_i) and edge e_i connecting vertices v_{i-1} and v_i. Then P is *monotone* if and only if, for every $i \in \{1, \ldots, n\}$, $|x_n - x_i| \leq |x_n - x_{i-1}|$ and $|y_n - y_i| \leq |y_n - y_{i-1}|$. Hence, a path in a graph embedded in the plane is monotone if and only if the Manhattan distance between its two terminal vertices equals the sum of the Manhattan lengths of its edges.

In a hypergraph, a *hyperpath* is a finite sequence of hyperedges in which each hyperedge intersects with its predecessor and successor. If the nodes (modules) of a hypergraph (netlist) are placed in the plane, the total length of a hyperpath is the sum of the HPWLs of its hyperedges (HPWL denotes minimum bounding-box half-perimeter). An edge $e = (v, w)$ is called the *equivalent edge* of a hyperedge h of a hypergraph in the plane, if (i) its vertices v and w are in h and (ii) e's minimum-HPWL bounding box is the same as h's minimum-HPWL bounding box. A hyperedge in the plane may have zero, one, or two equivalent edge(s).

A path P is called the *equivalent path* of a hyperpath H in a hypergraph in the plane, if there is a one-to-one correspondence between the edges of P and the hyperedges of H, such that every edge of P is the equivalent edge of its corresponding hyperedge in H. A *monotone chain* is a hyperpath which has an equivalent path that is monotone.

Assuming that no two vertices can occupy the same location, neighboring hyperedges in a monotone chain have exactly one vertex in common. These common vertices form the equivalent monotone path. The two terminal vertices of a monotone chain are the terminal vertices of its equivalent path. Hence, the length of a monotone chain equals the HPWL of the edge defined by the chain's two terminals.

Observation 2.1 *If the terminal vertices of a monotone chain $P = (h_1, h_2, ..., h_n)$ of a hypergraph G are fixed in the plane, there is no other planar embedding of hypergraph G which reduces the length of P.*

A *local net* is a hyperedge the HPWL of which is the minimum possible, subject to some spacing constraints between vertices. From Observation 2.1, it is evident that a placement has optimal HPWL if all its non-local nets can be partitioned into netwise-disjoint monotone chains with fixed endpoints.

2.2 The MC Algorithm

Starting from the placement of the real benchmark, sets of nets are identified that can be grouped together into netwise-disjoint monotone chains between well-separated fixed terminals. Local nets in the given placement are not modified. The main steps of the MC algorithm are sketched below.

Placement generation — The MC generator requires a placement of the original netlist. This placement is held fixed, while the netlist is changed so that the given placement attains the optimal HPWL for the modified netlist.

Net categorization — The nets of the original hypergraph are divided into three different categories depending on the placement of their pins:

(i) locally optimal-HPWL nets
(ii) nets that do not have equivalent edges
(iii) nets with equivalent edges.

Nets of Type (ii) cannot be members of monotone chains and are therefore modified. Nets of Type (iii) are labeled according to the directions of the monotone chains of which they can be members: from lower left toward upper right, or from lower right to upper left. Some of these nets can be members of chains in either direction.

Chain generation — As illustrated in Figure 1, sets of nets that can be members of the same chain are identified along with sets of empty regions that must later be filled by nets in order to complete the monotone chains. All nets of Type (iii) are assigned to chains during this step.

Chain removal — In our experiments, the number of empty regions created during chain generation is higher than the number of nets of Type (ii). Hence, to preserve netlist statistics, some of the chains generated are removed in order to reduce the number of empty regions and increase the number of available nets.

Gap covering — In the final step, empty regions between nets in chains are filled by new nets. Each new net replaces some available net in the original netlist. The new net includes the two pins defining the equivalent edge of the bounding box of its empty region $\mathcal{E}$ as well as additional pins selected from within $\mathcal{E}$ in order to match the degree of the replaced net.

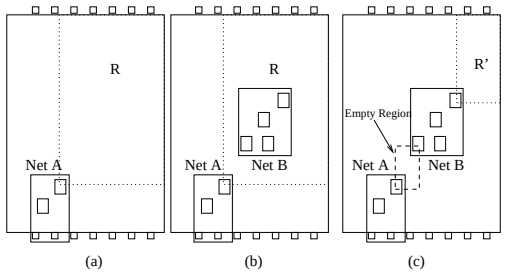

Figure 1: Example of chain generation. (a) Net A has already been added to the chain. A search for a new net takes place in region R. (b) Net B is selected to be added to the chain. (c) A gap is inserted between nets A and B, that will be covered later by a new net. A new search is initiated for nets in region R'.

Experiments reported in Section 4 suggest that, on the 2004 FastPlace-IBM standard-cell circuits with 20% white space, non-local nets probably do not represent a significant source of suboptimality for these tools. In order to amplify the suboptimality observed on mixed-size cases as much as possible, the PWS benchmarks described next include only local nets by default.

3 PWS Benchmark Construction

We refer to our placement suboptimality benchmarks with parametrized white space as PWS. As shown in Figure 2, the PWS generator produces a benchmark closely approximating

Set grid-resolution limit $\overline{N_G}$.

input

$N_\#$	target net-degree histogram
$P_{\rm mac}$	macro placement in core region $\mathcal{R}$
$N_{\rm sc}$	target number of standard cells
$\phi_{\rm ws}$	target white-space fraction

$\phi_{\rm mac} := \left(\sum_{v_i \in P_{\rm mac}} a(v_i)\right) / a(\mathcal{R})$.
$\phi_{\rm ws} := \min\{\phi_{\rm ws}, 1 - \phi_{\rm mac} - N_{\rm sc}/\overline{N_G}\}$.
$\phi_{\rm sc} := 1 - \phi_{\rm mac} - \phi_{\rm ws}$.
$N_G := N_{\rm sc}/\phi_{\rm sc}$.
if $(N_G > \overline{N_G})$ **then**
  $N_G := \overline{N_G}$; $\phi_{\rm sc} := N_{\rm sc}/N_G$; $\phi_{\rm ws} := 1 - \phi_{\rm mac} - \phi_{\rm sc}$.
end if
Snap $P_{\rm mac}$ into G, truncating macros as necessary;
  mark grid cells assigned to macros.
$N_{\rm ws} := \phi_{\rm ws} \cdot N_G$.
repeat
  Randomly select unvisited non-macro grid cell c.
  if (the spatial neighbors of c remain spatially
    connected in G when c is removed) **then**
    Mark c as white space and decrement $N_{\rm ws}$.
  end if
until ($N_{\rm ws} == 0$ **or**
      every non-macro grid cell has been examined)
if $(N_{\rm ws} > 0)$ **report failure and exit** **end if**
Mark all unmarked grid cells as standard cells;
  $\mathcal{V} := \{\text{macros}\} \cup \{\text{standard cells}\}$.
Following Figure 3, generate a minimal netlist
  "backbone" E_B, a connected set of local nets
  consistent with $N_\#$ which covers $\mathcal{V}$.
while ($N_\#$ still has nonzero entries **and**
      available locations for local nets still exist)
  Randomly select an available location p for a local net
  if (no new local net can be generated at p) **then**
    remove p from list of available local-net locations.
  else
    generate a local net of maximum possible degree k
    still represented in $N_\#$. Decrement $N_\#[k]$.
  end if
end while
output the placement suboptimality benchmark netlist

Figure 2: The PWS Benchmark Generator

the following four targets: (i) net-degree histogram $N_\#$, (ii) given placement $P_{\rm mac}$ of all macros, (iii) number of standard cells $N_{\rm sc}$, and (iv) white space fraction $\phi_{\rm ws}$. The ith component of vector $N_\#$ is the target number of nets of cardinality i. A macro is any module, fixed or movable, with height greater than the standard-cell row height. The generator places $N_{\rm sc}$ standard cells between macros and defines nets locally such that the total HPWL of the given placement is no more than a small, explicitly computed factor (1.00–1.08) above optimal for the final benchmark. Connectivity of the constructed netlist is ensured by inserting white space in such a way that all remaining cells and macros form a spatially connected set in the placement region.

As described in Figure 2 and below, the PWS generator proceeds in 4 stages:

1. Input target statistics; definition of uniform grid G; definition of mapping f_G which snaps a given macro placement $P_{\rm mac}$ into G.
2. Designation of white-space grid-cells, leaving cells and macros spatially connected.
3. Construction of the netlist backbone (Figure 3), a minimal connected set of local, near-optimal-HPWL nets connecting all cells and macros.
4. Construction of additional, optimal-HPWL local nets to match target netlist statistics as closely as possible.

An optional additional stage for the addition of optimal-HPWL *non-local* nets is described in Section 4.

Every legal mixed-size placement induces a complicated partition $\mathcal{R} = \mathcal{R}_{\rm mac} \cup \mathcal{R}_{\rm sc} \cup \mathcal{R}_{\rm ws}$ of its placement region $\mathcal{R}$ into three disconnected subregions: $\mathcal{R}_{\rm mac}$ occupied by macros, $\mathcal{R}_{\rm sc}$ by standard cells, and $\mathcal{R}_{\rm ws}$ left as white space. The PWS generator preserves a given macro placement $P_{\rm mac}$ precisely with respect to a fixed core region $\mathcal{R}$. Let $a(S)$ denote the area of subregion S. Region $\mathcal{R}$ is neither shrunk nor expanded relative to the macros — both $a(\mathcal{R})$ and $a(\mathcal{R}_{\rm mac})$ are held fixed. Instead, standard cells are uniformly shrunk or inflated to attain a higher or lower white-space targets, respectively. With this fixed-outline and fixed-macro-layout strategy, $\phi_{\rm mac} \equiv a(\mathcal{R}_{\rm mac})/a(\mathcal{R})$ is fixed, and it is evident that white space cannot be increased beyond the space left to it by the macros and standard cells:

$$\phi_{\rm ws} \le 1 - \phi_{\rm mac} - \phi_{\rm sc}^{\rm min},$$

where $\phi_{\rm sc}^{\rm min}$ denotes the minimum fraction of $\mathcal{R}$ which can be left for standard cells. The exact value of $\phi_{\rm sc}^{\rm min}$ is determined by storage and run-time considerations, as described next.

A tight lower bound on the optimal half-perimeter wirelength (HPWL) of each PWS benchmark is obtained by mapping the given macro layout $P_{\rm mac}$ into a uniform rectangular integer grid G of square cells over which all nets are defined. The mapping is denoted by $f_G : P_{\rm mac} \to 2^G$, where 2^G denotes the set of all subsets of grid cells in G. Each macro is identified by the mapping f_G with a distinct rectangular subset of grid cells in G. A nonoverlapping macro placement ensures that the grid-cell subsets associated with distinct macros are disjoint. Each center of each grid cell represents a candidate pin location. Pin locations on macros are restricted to grid-cells on macro boundaries and kept distinct. I.e., the center of each grid-cell along any macro's boundary can serve as a pin for at most one net. For simplicity, however, all pins on each standard cell are located at the same point at the center of that cell; i.e., the center of each standard cell may represent several pins for several different nets.

With all t pins of a given net placed at distinct grid-cell centers, the minimum HPWL of a t-pin net in such a grid is

input $\mathcal{C} := \emptyset$ = set of vertices contained in nets
$\mathcal{B} := \emptyset$ = set of vertices not yet in $\mathcal{C}$ but spatially
  adjacent (in G) to at least one vertex in $\mathcal{C}$.
Create a local net e at a random location.
Insert all $v \in e$ into $\mathcal{C}$ and all G-neighbors of e into $\mathcal{B}$.
while ($\mathcal{B}$ is not empty)
  Select an as yet unconnected grid cell $b \in \mathcal{B}$ and a
    connected grid cell $c \in \mathcal{C}$ such that b and c are
    adjacent in G. Cell b may be either a standard cell
    or a grid cell assigned to the boundary of an as yet
    unconnected macro. Cell c may be either a
    standard cell or an as yet unconnected grid cell
    assigned to the boundary of a connected macro.
  Create a net e containing b and c and containing as
    many other standard cells as possible, up to the
    maximum target net degree remaining in $N_\#$.
  if $(N_\#[|e|] > 0)$ **then** decrement $N_\#[|e|]$
  else
    $k := \min\{j \mid j > |e|$ **and** $N_\#[j] > 0\}$.
    decrement $N_\#[k]$ and increment $N_\#[k - |e|]$,
  end if
  for (all $v \in e$)
    remove v from $\mathcal{B}$ and insert it into $\mathcal{C}$.
    for (each grid neighbor w of v)
      if ($w \notin e$ **and** $w \notin \mathcal{C}$) **insert** w into $\mathcal{B}$ **end if**
    end for
  end for
end while
output minimal connected netlist E_B covering all $v \in \mathcal{V}$

Figure 3: PWS Netlist Backbone Generator

$r+s-2$, where $r = \lceil \sqrt{t} \rceil$ and $s = \lceil t/r \rceil$. This result is easily derived by packing the t square grid cells of the net into a rectangle of least possible perimeter. However, as shown in Figure 4, the optimal HPWL for a t-pin net may be attained by pin configurations with bounding boxes of different shapes.

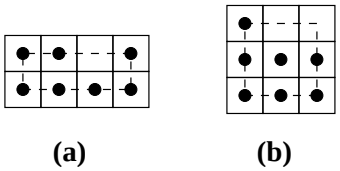

Figure 4: On a uniform square grid, the optimal HPWL of a 7-pin net (4 grid units) can be attained by pin configurations with either of the two bounding boxes shown as dashed line segments.

In order to construct a local net of optimal or near optimal HPWL containing a small subset of rectilinearly connected seed pin locations, rectangles of gradually increasing sizes containing the seeds are recursively examined. Each such rectangle is a rectangular subset of grid-cells containing the seed locations and representing a bounding box for a candidate net. In addition to the seeds, it may contain white space, standard cells, or grid-cells on the boundaries or interiors of macros. Of these, the available pin locations are the standard cells and the grid-cells on macro boundaries which have not yet been used as pins in other nets. As long as the number of available, rectilinearly connected pin locations in each such rectangle R is high enough to ensure optimal HPWL of the corresponding net, four larger rectangles containing R may also be considered. As shown in Figure 5, a rectangle is enlarged by adding to it a row or column of grid-cells along one of its four edges. Hence, the candidate rectangles for a given set of seeds form a quad-tree, the rectangles increasing in size along any path from root to leaf. Rectangles are enlarged until either optimal-HPWL cannot be obtained or the maximum-degree net remaining in $N_{\#}$ can be formed.

As is suggested by the labeling in Figure 5, incremental enumeration of distinct candidate optimal-HPWL bounding boxes amounts to the enumeration of distinct finite sequences $\{d_i\}_1^N$, where each $d_i \in \{n, s, e, w\}$ represents the direction of enlargement at the ith step, and $N = 1, 2, \ldots$ is the total number of enlargements for a given box. Two sequences of the same length N are distinct if and only if the numbers of occurrences of all the symbols $\{n, s, e, w\}$ are not the same for both. E.g., ns and sn are equivalent and lead to the same bounding box containing the initial seed box, but nse and nsn are distinct. The number of distinct sequences of length N is the number of ways $p_4(N)$ that the integer N can be expressed as the sum of 4 nonnegative integers; asymptotically, $p_4(N)$ grows with order $N^3/6$.[1] However, sequences for suboptimal bounding boxes (such as ree and rew in Figure 5) and their descendants can be easily avoided.

To further reduce search time, rectangles after a certain level in the quad-tree are enlarged in only one of the most promising directions, i.e., a direction containing the most available pin locations.

[1] The precise expression is $p_4(N) = (N^3 + 6N^2 + 11N + 6)/6$, which is the coefficient of x^N in the Taylor series for $(1-x)^{-4} = (1 + x + x^2 + x^3 + \cdots)^4$, assuming $|x| < 1$ [3].

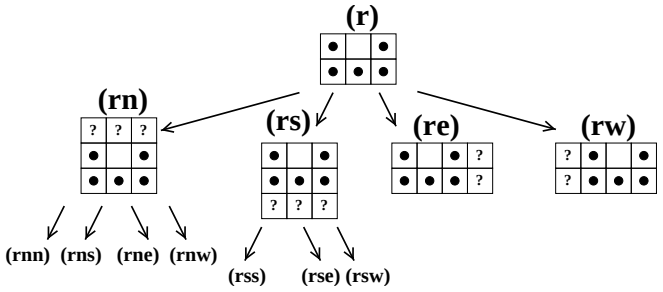

Figure 5: The first level of local search for the largest optimal-HPWL net containing a given 5-pin seed. After the first level, many duplicate (e.g., rsn) and suboptimal (e.g., ree) cases at the subsequent levels can be pruned.

Ideally, the resolution of grid G should be high enough to capture all macro and cell dimensions exactly. Our prototype implementation simplifies the definition of f_G in two ways. First, P_{mac} is represented in floating point; macro positions and dimensions are expressed as fractions of chip dimensions prior to their conversion to integer grid units. Macro dimensions are truncated in G as needed to snap macros into the grid. Because this change in units occasionally causes abutting macros in P_{mac} to overlap in G, a small fraction of macros may be discarded to prevent overlap. Second, each standard cell is represented by just one of the square grid cells of G — variations in standard-cell width are ignored. These two assumptions significantly reduce the size of G necessary to accurately represent P_{mac}. However, the resolution of G must still be large enough that
(i) each macro has nonzero height and width
(ii) the number of grid cells not used for macros is large enough to form both the requested number of standard cells N_{sc} and the requested fraction of white space ϕ_{ws}.

4 Experiments

Four sets of experiments with leading academic placement tools are reported. The first is on standard-cell MC circuits generated from the 2004 FastPlace-IBM benchmarks. The second is on mixed-size PWS circuits derived from the ISPD 2005 suite. The third considers the impact of introducing chains of optimal-HPWL nets into a PWS benchmark. The fourth examines the suboptimality of legalization and detailed-placement engines in isolation from their global-placement counterparts on a parametrized adaptation of the PWS circuits.

4.1 Non-local Nets (MC)

All MC benchmarks used in our experiments are generated from the FastPlace [7] versions of the 2002 IBM/ISPD benchmarks [2]. The white space in these test cases is approximately 20%. The FastPlace-IBM benchmarks modify the original IBM benchmarks by replacing macros with standard cells. However, the MC algorithm can also be applied to examples with macros for the generation of mixed-size circuits with known optimal placements. Although no new pads are explicitly inserted, most existing pads are connected to several nets each to allow for more chains.

The MC benchmark generator described in the previous section requires as input both a netlist and an initial "seed" place-

ment of that netlist. Dragon 3.01 [18] and mPL4 [9] were used to seed separate suites of MC benchmarks. Using other placers as seeds was observed to have negligible impact on final results, even when the placer used to create the seed placements was run on the resulting MC netlists.

The MC suite matches the FastPlace-IBM benchmarks exactly in number of cells, cell areas, number of nets, and net-degree distribution. Roughly 60–70% of the nets in the original and synthetic benchmarks are identical, and the distributions of net lengths in the optimal placement of the synthetic benchmarks are nearly identical to those of their seed placements on the original netlists. Moreover, the cell-degree distributions of the original and synthetic benchmarks are very similar (the *degree* of a cell is the number of nets containing the cell). Almost 80% of the cells in an MC netlist have a cell-degree difference at most 1 from their corresponding cells in the original netlist. Detailed statistics are shown in Figures 6 and 7.

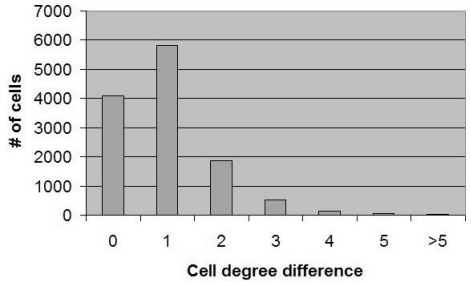

Figure 6: The cell-degree difference (in absolute values) distribution between the cells of mPL-MC01 and their corresponding cells in FastPlace-ibm01.

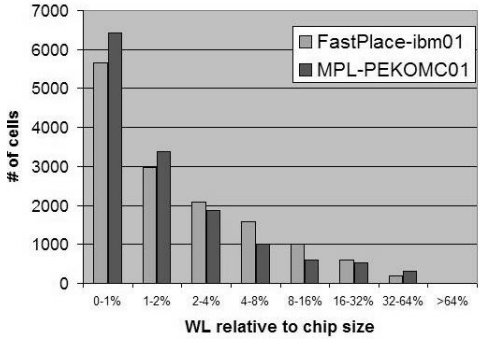

Figure 7: The wirelength distribution (relative to the chip half-perimeter) of the nets in FastPlace-ibm01 (as placed by mPL4) and the nets in mPL-MC01 (in their optimal placements).

Results for programs APlace 2.0 [14], mPL6-XDP [5, 10], and Capo 9.5 [1] on the Dragon-MC suite are shown in Figure 8. Very similar results (not shown) were obtained for all the tools on the mPL4-MC suite. The overall results show very good performance by all tools on all the benchmarks, regardless of which tool generates the initial placement used to seed the benchmark construction. The worst reported quality ratio by any of the placers on any benchmark is 1.07. We attribute this result to the increased range of optimal locations available to modules in multi-pin nets of monotone chains.

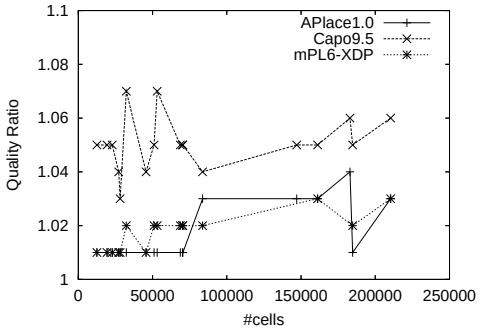

Figure 8: Results of some leading academic tools on MC Circuits seeded by Dragon 3.01 placements of FastPlace-IBM Benchmarks.

circuit	$\overline{N_{sc}}$	N_{mac}	$\overline{N_{nets}}$	ϕ_{mac}	ϕ^0_{ws}	ϕ^{max}_{ws}	ρ_{max}
PWS-A1	216180	63	233982	0.43	0.24	0.30	1.01
PWS-A2	264793	159	299358	0.62	0.21	0.29	1.03
PWS-A3	474287	723	531843	0.62	0.25	0.28	1.03
PWS-A4	531245	1329	563521	0.49	0.37	0.38	1.03
PWS-B1	280141	32	301577	0.17	0.46	0.46	1.01
PWS-B2	583514	23084	624625	0.38	0.38	0.40	1.03
PWS-B3	1137839	3778	1265913	0.67	0.14	0.24	1.08
PWS-B4	2237605	8170	2469988	0.38	0.35	0.40	1.03

$\overline{N_{sc}}$	average number of standard cells
N_{mac}	number of macros.
$\overline{N_{nets}}$	average number of nets.
ϕ_{mac}	macro-area utilization.
ϕ^0_{ws}	original benchmark's white-space fraction.
ϕ^{max}_{ws}	maximum white-space fraction attained by the generator.
ρ_{max}	maximum ratio of generated HPWL to its lower bound.

Table 1: PWS benchmark circuit statistics, with notation. Averages and maxima are taken over the 4 different white-space values by which each circuit is parametrized. Standard deviations of $\overline{N_{sc}}$ range from 0.5% to 4.2% of $\overline{N_{sc}}$; standard deviations of $\overline{N_{nets}}$ range from 10% to 16% of $\overline{N_{nets}}$.

4.2 Parametrized White Space (PWS)

The PWS approach gives the user control over the layout of the macros. In the ISPD 2005 benchmarks, all macro locations are prespecified for all circuits anyway, except `bigblue3`. For our construction based on `bigblue3`, we extracted movable macro locations from the placement generated for it by APlace [14] for the ISPD 2005 placement contest [20].

Each PWS local net's construction proceeds by depth-limited local search from a given subset of adjacent grid cells. A small amount of HPWL suboptimality is tolerated in some nets to simplify the implementation.[2] The optimal and attained HPWLs of the individual nets are simply added up to determine the limit on the total HPWL suboptimality in the final benchmark. These limits are shown in Table 1. On some circuits, nets e in the source netlist with more than a few hundred pins are represented by small subsets of high-degree nets whose pin counts sum to $|e|$.

[2] However, we still refer to the placements as optimal, because the set of modules in each net is rectilinearly connected and hence supports an optimal *routed* wirelength of the net.

	mPL6					APlace 2.0					Capo 9.5				
ckt\ ws	pack	5%	10%	20%	max	pack	5%	10%	20%	max	pack	5%	10%	20%	max
PWS-A1	1.80	1.35	1.48	1.70	1.80	1.33	1.50	1.22	1.15	1.54	6.17	3.33	3.14	3.05	2.67
PWS-A2	2.11	1.48	1.48	1.36	1.54	3.46	fail	fail	3.65	2.29	8.06	4.01	3.85	3.53	3.12
PWS-A3	4.32	2.14	1.52	1.41	1.33	2.27	1.23	1.14	1.13	1.10	4.10	2.10	1.93	1.53	1.33
PWS-A4	4.39	1.50	1.32	1.51	1.24	1.70	1.29	1.23	1.34	1.44	3.09	2.08	1.92	1.62	1.35
PWS-B1	—	1.30	1.34	1.30	1.24	—	1.44	1.32	1.17	1.33	—	2.50	2.42	2.08	1.77
PWS-B2	—	2.10	2.16	1.64	1.39	—	1.25	1.26	1.58	1.44	—	2.42	2.13	1.83	1.51
PWS-B3	—	1.54	1.62	1.99	2.02	—	2.14	1.59	2.02	2.23	—	2.49	2.10	1.89	1.91
PWS-B4	—	1.51	1.46	1.71	mem	—	1.26	1.21	1.16	1.33	—	mem	mem	mem	mem
Averages	3.16	1.61	1.55	1.58	1.51	2.19	1.45	1.28	1.65	1.59	5.35	2.70	2.50	2.22	1.96

Table 2: Results for mPL6, APlace 2.0, and Capo 9.5 on the PWS-ISPD2005 suboptimality benchmarks. Displayed are quality ratios of total computed HPWL to near-optimal HPWL upper bounds. Results with uniformly distributed white space are shown for 5%, 10%, 20%, and the maximum possible white space values. For PWS-adaptec1–4, quality ratios are also shown for benchmarks with optimal zero-white-space layouts ("pack") on the left side of the core region and 10% white space on the right. "mem" denotes an out-of-memory error. Capo 9.5 was run with option -noHMetis on PWS-a3, PWS-a4, and all four of the packed benchmarks; otherwise, all tools are in default mode in all cases.

Quality ratios of mPL6-XDP, APlace 2.0, and Capo 9.5 are listed in Table 2. The results show substantial variation both between tools and across different white-space values.

4.3 Suboptimality under both Parametrized White Space and Non-local Nets

The preceding results separate the impact of white space and mixed-size modules from that of non-local nets. However, the MC and PWS techniques can be combined into a single set of suboptimality benchmarks supporting parametrized percentages of both non-local nets and white space. A combination derived from the PWS construction (Figure 2) was tested on the mixed-size IBM01 benchmark from the ICCAD2004 test suite [1], as follows. Following the construction of the PWS netlist backbone (Figure 3), monotone chains of non-local nets are constructed as follows.

1. The set of all boundary pads and candidate pin locations of fixed macros is partitioned by a simple heuristic into pairs of fixed terminals, such that the terminals in each pair are relatively far apart.
2. For each pair of terminals, designate one terminal in the pair as the start, and another as the end. A chain of non-local nets is iteratively constructed for the pair of terminals by the following sequence of steps (compare to Figure 1).
   (a) Randomly select an available pin location in the bounding box of the end terminal and the net corner pin most recently added to the the chain. The selected location is the next net corner pin.
   (b) Randomly select additional pins in the resulting bounding box of that new net-corner pin location and the preceding net-box corner pin to populate the net.

   Net-box corner-pin locations are selected at randomized distances from one another approximately 1/10 of the width or height of the placement region, until the end terminal of the chain is reached.

Results of APlace 2.0, Capo 10, and mPL6, all run in default mode, are shown for the combined PWSMC IBM01 benchmark in Table 3, both without and with non-local nets. Macros larger than 10 cell rows high were treated as fixed, their boundaries thus supplying some additional terminal locations. As expected, the presence of monotone chains of non-local nets decreases all placers' suboptimality ratios.

With Local Nets Only			
	APlace	Capo	mPL
ibm01-10WS	1.20	1.88	1.31
ibm01-40WS	1.40	1.96	1.27
Averages	1.30	1.92	1.29

With Chains of Optimal-hpwl Non-local Nets					
	$\frac{\#nln}{\#nets}$	$\frac{WL_{nonloc}}{WL_{total}}$	APlace	Capo	mPL
ibm01-10WS	0.15	0.57	1.11	1.49	1.16
ibm01-40WS	0.13	0.68	1.08	1.67	1.10
		Averages	1.10	1.58	1.13

Table 3: HPWL Suboptimality of APlace 2.0, Capo 10, and mPL6, compared on 10% and 40% white-space versions of a PWS circuit derived from the ICCAD 2004 IBM01 mixed-size benchmark, both without (top) and with (bottom) the addition of optimal-HPWL nonlocal nets. Approximately 13,15% of the nets in the second set are nonlocal, accounting for 57%,68% of total HPWL.

4.4 Suboptimality of Detailed Placement

Optimal global placements (OGP) parametrized by bin size were generated from the optimal PWS placements as follows. Uniform rectangular bin grids of user-specified dimensions were superimposed. Cells and macros centered in the same bin were moved to the bin center, where they were placed concentrically. These OGP placements were then used as benchmarks for detailed placement by mPL6-XDP [5, 10] and APlaceDP [14]. Each PWS circuit can generate several different OGP circuits, one for each bin size. The DP engines were run on a set of these OGP circuits, and the rate of degradation in their quality with respect to bin size and white-space value was observed. For each of the different white-space values, the quality ratios obtained by the DP engines were averaged over the 8 different circuits. The result is illustrated in Figure 9. The benchmarks reveal opposite trends in these engines with respect to increasing white space. For these test cases, XDP's performance degrades as white space increases, while APlaceDP's improves. APlace's cell-swapping strategy may have some advantage on these benchmarks, because the standard cells in these test cases are all of uniform size and shape. Under higher white space, the size of the set of candidate swaps is reduced, making successful swaps more likely to

be found. On the other hand, XDP's local-window-based refinement is apparently a drawback on the higher-white-space cases, where larger scale moves are apparently needed.

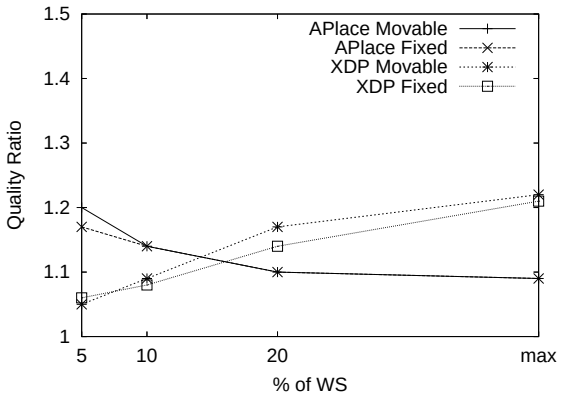

Figure 9: Average quality ratios of APlaceDP and XDP over the 8 different netlists of OGP DP benchmarks.

Results on the OGP benchmark derived from the PWS-`adaptec2` benchmark with 10% uniformly distributed white space are summarized in Figure 10. Results are shown for two scenarios: one in which all macros are held fixed, and hence only standard cells are aggregated into bin centers, and another in which both kinds of objects are moved from their locations in the known near-optimal placement to the nearest bin center. The results of these experiments show that the quality of detailed placement deteriorates fairly rapidly as the bin size increases, even for these uniformly accurate global placements. For bin sizes up to 4×4, macro legalization is not a primary source of suboptimality, but for larger bin sizes, it is.

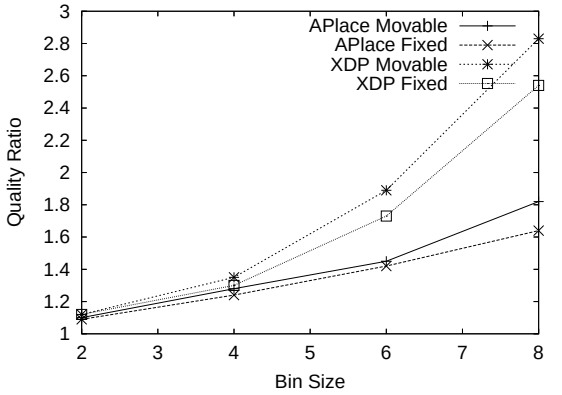

Figure 10: Suboptimality of APlaceDP and XDP on the OGP DP benchmarks generated from PWS-adaptec2 with 10% white space.

The OGP benchmarks provide a means of estimating how much of a placer's suboptimality is attributable to its global placement, and how much to legalization and detailed placement. Table 4 compares the suboptimality observed for XDP on 2×2 OGP cases to that observed for mPL6-XDP, including both GP and XDP, on the corresponding PWS source circuits from which the OGP cases are derived. Subtracting the observed XDP suboptimality on the 2×2 OGP benchmark from the total mPL6-XDP $\equiv$ mPL6GP+XDP suboptimality on the corresponding PWS benchmark gives an estimate of the mPL6 GP suboptimality. It should be noted, however, that these suboptimality values are not truly additive, for at least two reasons. First, the starting configuration for XDP on the OGP benchmark is very different from its starting configuration on the corresponding PWS benchmark. Second, relative module positions in a global placement below the resolution of the 2×2 OGP grid will typically be used as hints during legalization and DP to improve results.

%WS	GP	DP	Total
5	51%	10%	61%
10	46%	9%	55%
20	39%	17%	56%
40	29%	22%	51%

Table 4: Estimated suboptimality of mPL6 global (GP) and detailed placement (DP) engines. The GP estimate is obtained simply by subtracting the observed DP quality ratio obtained on the 2×2 OGP benchmark from the overall quality ratio observed for the corresponding PWS benchmark.

Figure 11 displays line segments between modules' placed locations and their optimal locations in a small PWS testcase (IBM02) constructed with 5% white space from the ICCAD2004 mixed-size suite [1]. Results for both global and detailed placements of APlace 2.0 and mPL6-XDP are shown. From these plots, it is clear that displacement errors are not at all randomly distributed, and, on the contrary, display large-scale systematic bias. We observe similar trends in displacement plots for other tools on other PWS circuits and at other white-space fractions. We conclude that, even when each cell in a global placement is very close to (one of) its optimal location(s), further reduction in the objective can often only be achieved by moving large subsets of cells simultaneously by small amounts. Iterative, local, window-based refinement will not remove the systematic error.

5 Conclusions

Two new sets of synthetic benchmark circuits with known optimal-HPWL or near-optimal-HPWL placements have been presented. The MC set quantifies the role of non-local nets in suboptimality; the PWS set quantifies the role of white space and modules of mixed size. Experiments with leading academic placement tools support three main conclusions. First, as shown in Table 2, different tools produce widely varying results on some of the mixed-size PWS benchmarks. Hence, these benchmarks can be used to identify deficiencies in tools producing relative poor results. Second, the presence of netwise-disjoint chains of nets linking pairs of numerous, well-distributed, fixed terminals appears to make wirelength-driven placement by contemporary methods considerably less difficult. Third, the accumulation of small but systematic errors in the placement of local nets appears to be a greater source of suboptimality than the total error in identifying and placing non-local nets. The corrective action needed to further reduce that suboptimality, whether taken during global placement, legalization, or detailed placement, must consider simultaneous motion of large subsets of objects in order to be effective. Restriction to subsets localized in an arbitrary way is, in general, insufficient to improve on existing results.

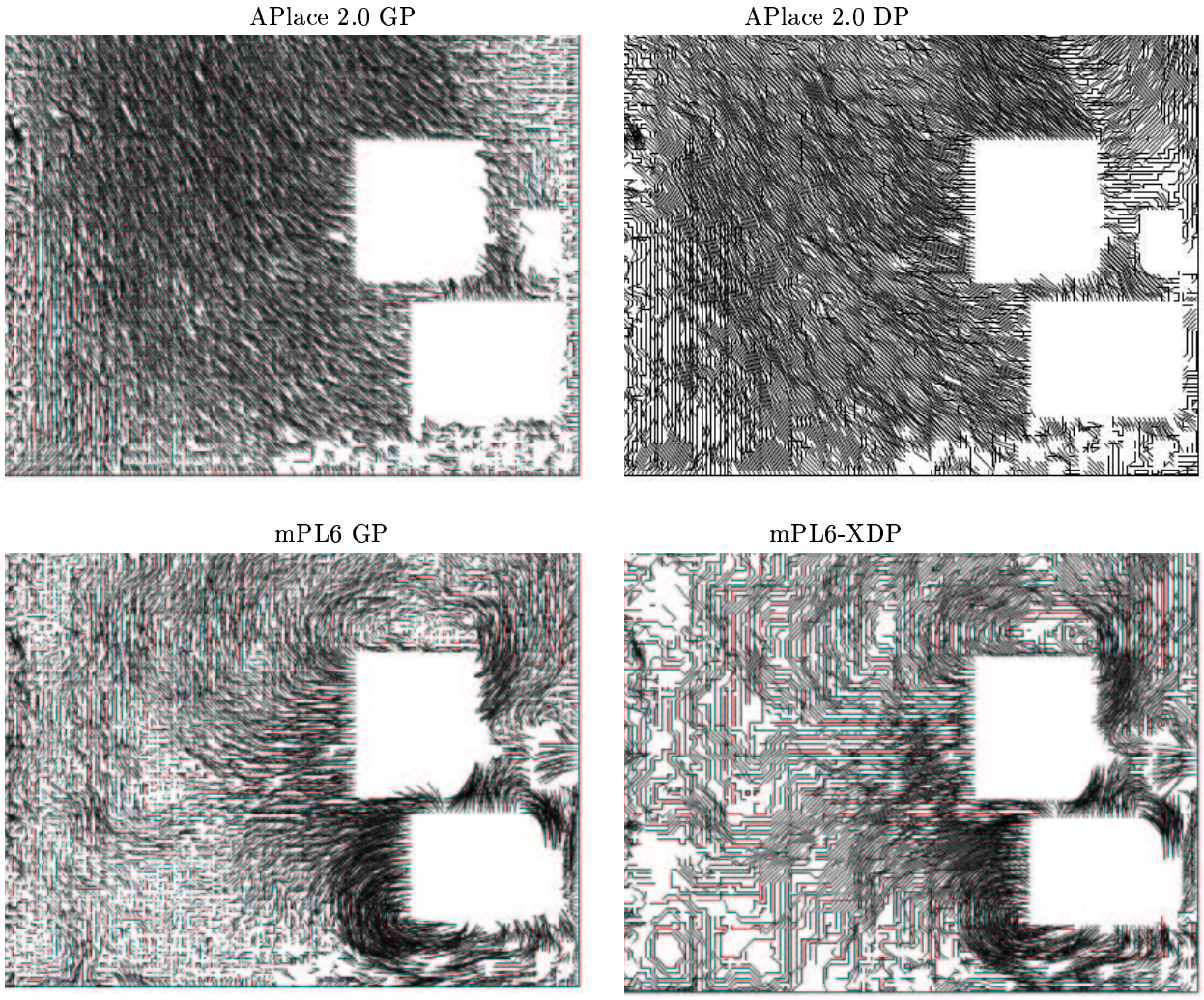

Figure 11: Individual module displacements from optimal on the PWS-ICCAD04-IBM02 benchmark with 5% white space. Displacements of both global and detailed placements are shown for APlace (top) and mPL6-XDP (bottom). The HPWL quality ratios observed on this benchmark are 1.45 for APlace and 1.23 for mPL.

References

[1] S. Adya, S. Chaturvedi, J. Roy, D. Papa, and I. Markov. Unification of partitioning, placement and floorplanning. In *Proc. Int'l Conf. on Comp.-Aided Design*, pages 12–17, 2004.

[2] C. Alpert. The ISPD98 Circuit Benchmark Suite. In *Proc. Int'l Symp. on Phys. Design*, pages 80–85, 1998.

[3] G. Andrews and K. Eriksson. *Integer Partitions*. Cambridge University Press, 2004.

[4] U. Brenner and M. Struzyna. Faster and better global placement by a new transportation algorithm. In *Proc. Design Automation Conf.*, pages 591–596, 2005.

[5] T. Chan, J. Cong, and K. Sze. Multilevel generalized force-directed method for circuit placement. In *Proc. Int'l Symp. on Phys. Design*, 2005.

[6] C. Chang, J. Cong, M. Romesis, and M. Xie. Optimality and scalability study of existing placement algorithms. *IEEE Trans. on Comp.-Aided Design of Integrated Circuits and Sys.*, pages 537–549, 2004.

[7] C. Chu and N. Viswanathan. FastPlace: Efficient analytical placement using cell shifting, iterative local refinement, and a hybrid net model. In *Proc. Int'l Symp. on Phys. Design*, pages 26–33, April 2004.

[8] J. Cong, M. Romesis, and M. Xie. Optimality, scalability and stability study of partitioning and placement algorithms. In *Proc. International Symposium on Physical Design*, 2003.

[9] J. Cong, J. R. Shinnerl, M. Xie, T. Kong, and X. Yuan. Large-scale circuit placement. *ACM Trans. on Design Automation of Electronic Systems*, 10(2):389–430, 2005.

[10] J. Cong and M. Xie. A robust detailed placement for mixed-size IC designs. In *Proc. Asia South Pacific Design Automation Conf.*, pages 188–194, 2006.

[11] R. Goering. Placement tools criticized for hampering IC designs. *EE Times*, 2003. http://www.eedesign.com/story/OEG20030205S0014.

[12] L. Hagen, D. J.-H. Huang, and A. Kahng. Quantified suboptimality of VLSI layout heuristics. In *Proc. Design Automation Conference*, pages 216–221, 1995.

[13] A. Kahng and S. Reda. Evaluation of placer suboptimality via zero-change netlist transformations. In *Proc. Int'l Symp. on Phys. Design*, pages 208–215, Apr 2005.

[14] A. Kahng, S. Reda, and Q. Wang. Architecture and details of a high quality, large-scale analytical placer. In *Proc. Int'l Conf. on Comp.-Aided Design*, November 2005.

[15] Q. Liu and M. Marek-Sadowska. A study of netlist structure and placement efficiency. In *Proc. Int'l Symp. on Phys. Design*, pages 198–203, 2004.

[16] G.-J. Nam, C. Alpert, P. Villarrubia, B. Winter, and M. Yildiz. The ISPD2005 placement contest and benchmark suite. In *Proc. Int'l Symp. on Phys. Design*, pages 216–220, Apr 2005.

[17] S. Ono and P. Madden. On structure and suboptimality in placement. In *Proc. Asia South Pacific Design Automation Conf.*, Jan 2005.

[18] M. Sarrafzadeh, M. Wang, and X. Yang. *Modern Placement Techiques*. Kluwer, Boston, 2002.

[19] Q. Wang, D. Jariwala, and J. Lillis. A study of tighter lower bounds in LP relaxation based placement. In *ACM Great Lakes Symposium on VLSI*, pages 498–502, 2005.

[20] http://www.sigda.org/ispd2005/contest.htm.