Optimality and Scalability Study of Existing Placement Algorithms

Chin-Chih Chang, Jason Cong, *Fellow, IEEE*, Michail Romesis, *Student Member, IEEE*, and Min Xie

***Abstract*—Placement is an important step in the overall IC design process in deep submicron technologies, as it defines the on-chip interconnects which have become the bottleneck in determining circuit performance. The rapidly increasing design complexity, combined with the demand for the capability of handling nearly flattened designs for physical hierarchy generation, poses significant challenges to existing placement algorithms. There are very few studies dedicated to understanding the optimality (i.e., the comparison of the solution of an algorithm to the optimal solution) and scalability (i.e., the analysis of the degradation of the performance of an algorithm as the input size of the problem increases) of placement algorithms, due to the limited sizes of existing benchmarks and limited knowledge of optimal solutions. The contribution of this work includes three parts. 1) We implemented an algorithm [Placement Examples with Known Optimal (PEKO) algorithm] for generating synthetic benchmarks that have known optimal wirelengths and can match any given net degree distribution profile. 2) Using benchmarks of 10 k to 2 M placeable modules with known optimal solutions, we studied the optimality and scalability of four state-of-the-art placers, Dragon (Wang *et al.*, 2000), Capo (Caldwell *et al.*, 2000), mPL (Chan et al., 2000), and mPG (Chang et al., 2002) from academia, and a leading edge industrial placer, QPlace (Cadence 1999) from Cadence. For the first time our study reveals the gap between the results produced by these tools versus true optimal solutions. The wirelengths produced by these tools are 1.59 to 2.40 times the optimal in the worst cases, and are 1.43 to 2.12 times the optimal on average. As for scalability, the average solution quality of each tool deteriorates by an additional 9% to 17% when the problem size increases by a factor of ten. These results indicate significant room for improvement in existing placement algorithms. 3) We studied the impact of nonlocal nets on the quality of the placers by extending the PEKO algorithm (PEKU algorithm) to generate synthetic placement benchmarks with a known upper bound of the optimal wirelength. For these benchmarks, the wirelengths produced by these tools are 1.75 to 2.18 times the wirelength upper bound in the worst case, and are 1.62 to 2.07 times the wirelength upper bound on average. Moreover, in our study we found that the effectiveness of the algorithms varies for circuits with different characteristics.**

***Index Terms*—Deep submicron (DSM), optimization, physical design, placement.**

Manuscript received May 31, 2003; revised September 17, 2003 and December 4, 2003. This work was supported in part by Semiconductor Research Corporation under Contracts 98-TJ-686 and 2003-TJ-1019, in part by the National Science Foundation under Grant CCR–0096383, and in part by DARPA/GSRC under Contract SA2211-23106. This paper was recommended by Guest Editor C. J. Alpert.

C.-C. Chang was with the Computer Science Department, University of California, Los Angeles, CA 90095 USA. He is now with Cadence Design Systems, Inc., San Jose, CA 95134 USA (e-mail:chinchih@cadence.com).

J. Cong, M. Romesis, and M. Xie are with the Computer Science Department, University of California, Los Angeles, CA 90095 USA (e-mail: cong@cs.ucla.edu; michail@cs.ucla.edu; xie@cs.ucla.edu).

Digital Object Identifier 10.1109/TCAD.2004.825870

I. Introduction

PLACEMENT is an important step in the overall IC design process in deep submicron (DSM) technologies, as it defines the on-chip interconnects, which have become the bottleneck in determining circuit performance.

According to the ITRS'01 Roadmap [6], the maximum number of transistors per chip will be over 1.6 billion, with a clock frequency of 28.7 GHz by the year 2016. Such high complexity poses significant challenges to the scalability of placement algorithms. The traditional way to handle large designs is through partitioning according to the logical hierarchy. However, it is pointed out in [7] that these hierarchies are derived with little or no consideration for the physical layout and they may not embed well in a two-dimensional silicon surface. Therefore, it is proposed in [7] that the right way to partition the design is to first flatten the logic hierarchy to the extent that we are certain about the "physical locality" of each module in the flattened design, and then construct a physical hierarchy (coarse placement) on this nearly flattened netlist. The algorithm presented in [4] is developed to support this methodology. In general, this approach requires highly scalable placement algorithms which can handle nearly flattened designs with 100 k to 10 M placeable objects.

Various algorithms have been proposed in the past 30 years, including min-cut-based methods [2], analytical methods [8], and iterative methods [9]. Direct comparison of placement algorithms is usually difficult [10], [11] because placers make different assumptions and use different test cases. By summarizing published results, we found the rate of wirelength reduction to be only 5%–10% every two to three years since the 1980s. In 1988, Gordian [12] reported substantial wirelength reduction over its predecessors. In 1991, Gordian-L [8] reported a 20% wirelength reduction over Gordian. TimberWolf 7.0 [13] reduced Gordian's wirelength by 10% in 1993. The iterative force-directed method [14] outperformed Gordian-L in 1998 by an average of 6%. mPL [3] runs 10 $\times$ faster than Gordian-L with a penalty of wirelength increase of 10%. The latest developments in placement algorithms in the past three years, including Capo [2], Dragon [1], Mongrel [15], and mPG [4] vary mostly in runtime. The wirelength difference between Dragon and Capo is within 5% [16], but Dragon is 7 $\times$ slower. mPG is about 2 $\times$ faster than Dragon with a wirelength that is up to 5% longer [17]. Mongrel's wirelength is also slightly worse than Dragon's [18]. The lack of significant progress prompts us to wonder whether there remains much room for improvement in circuit placement (at least in terms of wirelength minimization).

Until now, there have been few studies dedicated to understanding the optimality and scalability of placement algorithms.

This is due to the limited sizes of existing benchmarks and limited knowledge of their optimal solutions. Two types of benchmarks are commonly used. One type of benchmark is based on real designs [19]–[21]. They are either directly extracted from real designs [19], or based on minor perturbations of real designs [20], [21]. Another type of benchmark is synthetic, i.e., circuits generated by computer programs. Several algorithms [22]–[25] can generate benchmarks with the user-specified Rent's parameter [26]. Other possible inputs to the generation algorithms include design size, net degree distribution, logic functionality, etc. As an application of synthetic benchmarks, [27] used benchmarks from [23] to search Rent's parameter that incurred the highest resource utilization ratio. The study in [28] attempted to quantify the suboptimality of placement algorithms in terms of chip area by "stitching" small designs to form large designs. The study in [29] showed that in datapath layouts, the area of automated standard cell-based designs can be $14\times$ larger than custom designs. The major drawback shared by these studies is that the optimal solutions for placement are unknown. It is difficult to determine how the solution quality changes as the design size grows.

The contribution of this work includes three parts. 1) We implemented an algorithm [Placement Examples with Known Optimal (PEKO) algorithm] for generating synthetic benchmarks that have known optimal wirelengths and can match any given net degree distribution profile. Our algorithm is similar to the one first proposed by Boese, which was outlined in [28].[1] 2) Using benchmarks of 10 k to 2 M placeable modules with known optimal solutions, we experimented with four state-of-the-art placers from academia, Dragon [1], Capo [3], mPL [9] and mPG [4], and a leading edge industrial placer, QPlace [5] from Cadence. For the first time our study reveals the gap between the results produced by these tools versus true optimal solutions. The wirelengths produced by these tools are 1.59 to 2.40 times the optimal in the worst cases, and are 1.43 to 2.12 times the optimal on the average. As for scalability, the average solution quality of each tool deteriorates by an additional 9% to 17% when the problem size increases by a factor of 10. These results indicate significant room for improvement in existing placement algorithms. 3) We studied the impact of nonlocal nets on the quality of the placers by extending the PEKO algorithm (PEKU algorithm) to generate synthetic placement benchmarks with a known upper bound of the optimal wirelength. Even for these benchmarks, the wirelengths produced by these tools are 1.76 to 2.18 times the wirelength upper bound in the worst case, and are 1.62 to 2.07 times the wirelength upper bound on average. Furthermore, none of the placers produce consistently better results than the others when the percentage of nonlocal nets goes from 0.25% to 10%. The preliminary results were published in [31] and [32], and covered as feature stories in the *Electrical Engineering Times* magazine in February [33] and April [34], 2003. These results have generated great interest among the industrial designers and academic researchers, and over 60 downloads of the PEKO and PEKU test suites by major universities and EDA and semiconductor companies, e.g., Cadence, Synopsys, Magma, IBM, and Intel, etc.

[1]Boese, however, never implemented his idea nor experimented it with any placer [30].

The rest of this paper is organized as follows: Section II describes the PEKO benchmark generation algorithm. Section III describes the PEKU benchmark generation algorithm. Section IV presents experimental results for the synthetic benchmarks. Section V gives conclusions and future work.

II. Placement Benchmark Generation With Known Optimal Wirelength

The optimal placement solutions for real circuits are usually unknown. However, for our optimality study, we can construct a circuit with known optimal wirelength using the characteristics of a real circuit, and measure the solution quality of existing placement algorithms on these circuits.

A. Problem Formulation

First, we introduce some notations: Given a netlist N, let p be the number of placeable modules in the netlist, and let $D(N) = (d_2, d_3, \ldots d_n)$ be the *Net Distribution Vector* (*NDV*), where d_k is the total number of k pin nets in the netlist.

We would like to solve the following problem: Given a number p and a vector D, construct a placement benchmark with p placeable modules, such that its netlist has D as its *NDV* and has a known optimal half-perimeter wirelength solution.

B. PEKO Algorithm

1) Algorithm Description: Our algorithm, PEKO, makes two assumptions: all the modules are of equal size, and there is no space between the rows. It first places all the modules in a rectangular region close to a square, then connects the nets to the modules one-by-one, using the minimum perimeter bounding box for each net. In the end, a netlist is extracted from this placed configuration.[2] Table I gives a description of the algorithm.

Fig. 1 shows an example when $p = 9$, $D = (6, 2, 2)$. Net A is a four-pin net. According to our algorithm, it will connect four modules located in a 2×2 rectangular region. In Fig. 1, it connects the four modules in the lower left corner. The other four-pin net B is placed on the lower right corner. Similarly, the two three-pin nets are generated as C and D, respectively. This process is repeated until the *NDV* is exhausted. The total wirelength for this benchmark is $6*1+2*2+2*2 = 14$.

2) Proof of Optimality: According to the generation algorithm, the wire length of each k-pin net is $\lceil\sqrt{k}\rceil + \lceil k/\lceil\sqrt{k}\rceil\rceil - 2$. For any k-pin net, the optimal half perimeter wire length can only be achieved when the modules of this net are placed in a rectangular region close to a square, i.e., the length of each side is close to $\lceil\sqrt{k}\rceil$. In particular, the width and height of the rectangle should be $\lceil\sqrt{k}\rceil$ and $\lceil k/\lceil\sqrt{k}\rceil\rceil$, respectively (or $\lceil k/\lceil\sqrt{k}\rceil\rceil$ and $\lceil\sqrt{k}\rceil$). The wirelength of such a configuration is $\lceil\sqrt{k}\rceil + \lceil k/\lceil\sqrt{k}\rceil\rceil - 2$. The wirelength of an n-pin net

[2]It is not explicitly checked if the netlist is connected. When the number of nets is far less than that of cells, the netlist may have disconnected components. However, the net profile we used always have comparable number of nets and cells. Furthermore, our method picks the cell with the lowest number of connections each time. As a result, the generated netlist is usually connected.

TABLE I
ALGORITHM FOR PLACEMENT BENCHMARK GENERATION

PEKO Algorithm	
Input	Total number of modules p Net Distribution Vector $D = (d_2, d_3, \ldots d_n)$
Output	Placement Benchmark having D as its *NDV* with known optimal wirelength

Put all the p modules in a $\lceil\sqrt{p}\rceil \times \lceil p/\lceil\sqrt{p}\rceil\rceil$ shaped region
$j \leftarrow 0$
For $i \leftarrow n$ to 2 **Do**
  While $d_i > 0$ **Do**
    Randomly pick a module m which has the least number of nets connected
    Randomly select a bounding box b that includes m and has size $\lceil\sqrt{i}\rceil \times \lceil i/\lceil\sqrt{i}\rceil\rceil$ (or $\lceil i/\lceil\sqrt{i}\rceil\rceil \times \lceil\sqrt{i}\rceil$)
    Generate net_j connecting i modules in b
    $d_i \leftarrow d_i$-1
    $j \leftarrow j$+1
  End While
End For
Output design size and dimension
For $i \leftarrow 0$ to $j-1$ **Do**
  Output the modules connected by net_i
End For

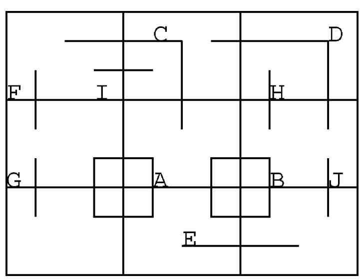

Fig. 1. Benchmark generation for $p = 9, D = (6, 2, 2)$.

achieved by our algorithm is optimal, and the total wirelength is the sum of all the nets; therefore, it is also optimal.

Given a benchmark E generated by PEKO with *NDV* D, $\sum_{i=2}^{n} d_i * (\lceil\sqrt{i}\rceil + \lceil i/\lceil\sqrt{i}\rceil\rceil - 2)$ is the optimal wirelength of the benchmark, denoted as $OW(E)$. Given a placement solution s to benchmark E, we measure its wirelength and denote it as $PW_s(E)$. We define the ratio $PW_s(E)/OW(E)$ as the wirelength ratio (WR) of placement solutions. This metric gives us an objective evaluation of a solution.

C. Generation of a "Realistic" Benchmark Set With Known Optimal Wirelength

In order to generate realistic benchmarks, we first extract the module numbers and *NDV*s from the netlists in the ISPD'98 suite [19] (originally from IBM) and generate a set of benchmarks named suite-1 using PEKO. Table II gives the characteristics of suite-1. The column "OW" gives the optimal half-perimeter wirelength for each benchmark. Suite-2 is generated by scaling the module number and *NDV* of each circuit in suite-1 by a factor of ten.

One important feature of suite-1 and suite-2 is that there is no net connected with pads. This feature is enforced from the concern that such nets may give a hint about where to place each net. To make our study complete, we also generate two more sets of benchmarks that have nets connected with pads, since some analytical placement algorithms make use of fixed pad locations to avoid degenerate solutions. The generation of the pad-connected nets is as follows. A pad is randomly picked on the boundary. Then a rectangular region abutting it is determined. The dimension of this region is calculated from the degree of the net. A new net is constructed by connecting the cells in this region with the pad. It is obvious that the wirelength for such a net is still optimal. These circuits are named suite-3 and suite-4, respectively. Table III gives a description of suite-3. All four suites are given in both GSRC BookShelf format and LEF/DEF format, and can be downloaded from [35].

TABLE II
CHARACTERISTICS OF SUITE-1 (SUITE-2 IS GENERATED BY SCALING THE MODULE NUMBER AND NDV OF EACH CIRCUIT IN SUITE-1 BY A FACTOR OF 10)

circuit	#cell	#net	#row	OW
Peko01	12506	13865	113	8.14E+05
Peko02	19342	19325	140	1.26E+06
Peko03	22853	27118	152	1.50E+06
Peko04	27220	31683	166	1.75E+06
Peko05	28146	27777	169	1.91E+06
Peko06	32332	34660	181	2.06E+06
Peko07	45639	47830	215	2.88E+06
Peko08	51023	50227	227	3.14E+06
Peko09	53110	60617	231	3.64E+06
Peko10	68685	74452	263	4.73E+06
Peko11	70152	81048	266	4.71E+06
Peko12	70439	76603	266	5.00E+06
Peko13	83709	99176	290	5.87E+06
Peko14	147088	152255	385	9.01E+06
Peko15	161187	186225	402	1.15E+07
Peko16	182980	189544	429	1.25E+07
Peko17	184752	188838	431	1.34E+07
Peko18	210341	201648	460	1.32E+07

TABLE III
CHARACTERISTICS OF SUITE-3 (SUITE-4 IS GENERATED BY SCALING THE MODULE NUMBER AND NDV OF EACH CIRCUIT IN SUITE-3 BY A FACTOR OF 10)

circuit	#cell	#net	#row	OW
Peko01	12506	14111	113	8.22E+05
Peko02	19342	19584	140	1.27E+06
Peko03	22853	27401	152	1.51E+06
Peko04	27220	31970	166	1.76E+06
Peko05	28146	28446	169	1.95E+06
Peko06	32332	34826	181	2.07E+06
Peko07	45639	48117	215	2.89E+06
Peko08	51023	50513	227	3.15E+06
Peko09	53110	60902	231	3.65E+06
Peko10	68685	75196	263	4.75E+06
Peko11	70152	81454	266	4.72E+06
Peko12	70439	77240	266	5.02E+06
Peko13	83709	99666	290	5.89E+06
Peko14	147088	152772	385	9.03E+06
Peko15	161187	186608	402	1.16E+07
Peko16	182980	190048	429	1.25E+07
Peko17	184752	189581	431	1.35E+07
Peko18	210341	201920	460	1.32E+07

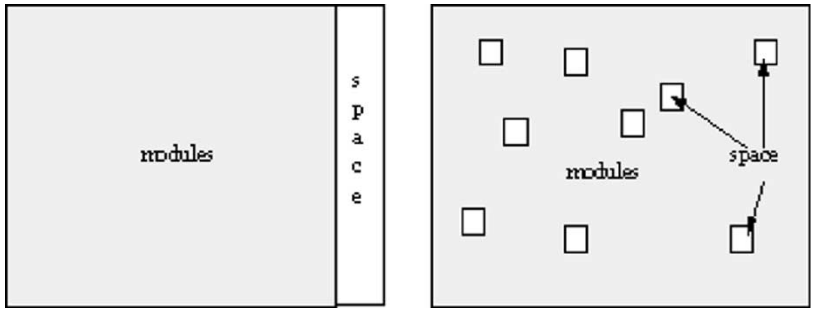

Fig. 2. White space generation methods.

D. White Space Generation

To further mimic real designs, we take a simplistic approach to generate white space in the PEKO suite. After the optimal configuration is obtained, white space is inserted to the right of the placeable modules. For each circuit in PEKO, 15% of the chip area is white space.[3]

An alternative is to first connect each module with at least one net, then randomly remove $\alpha \times p$ modules and all the nets connected with them, where α is the ratio of desired space area to the chip area, as shown in Fig. 2. It is easy to prove that benchmarks thus generated also have a known optimal wirelength. Furthermore, the white space is randomly distributed on the chip. This method, however, may not give a benchmark matching the desired *NDV*. Therefore, it is not used for our benchmark generation.

III. Placement Benchmark Generation With Global Connections

The generation of the PEKO suite suffers from one drawback. In the optimal solutions, all the nets are local, i.e., their wirelength is the minimum possible value. This may not be true in real circuits, which may also have global connections that span a significant portion of the chip, even when they are optimally placed. Table IV gives the profile of placed results of the ISPD'98 benchmarks produced by Dragon. The second and third columns are the width and height of the chip, respectively. The fourth column gives the wirelength of the longest net in each circuit. The last column gives the percentage of wirelength contributed by the longest 10% of the total nets. It can be seen that even for a small number of global connections, their wirelength contribution is significant. Therefore, the performance of a placer can be quite different due to the presence of these global nets. It is worthwhile to study the performance of different placers in the presence of global nets.

We take two approaches to consider the impact of global nets. One is to generate circuits consisting only of global nets; the other is to introduce some randomly generated, nonlocal nets into the PEKO suite. We use the term "nonlocal net" to denote the nets in a placement solution whose wirelength is larger than the minimum possible value.

[3]The initial circuits had no white space. However, the wirelength produced by some placers are abnormally longer than the optima because of the tight area constraint. To relieve this issue, we inserted 15% white space.

TABLE IV
PROFILE OF PLACEMENT RESULTS BY DRAGON [1] ON ISPD'98

circuit	height	width	WL of longest net	WL of longest 10%
ibm01	8158	4530	7148	51%
ibm02	8158	6430	14224	46%
ibm03	8158	6740	10624	58%
ibm04	8158	9140	15171	53%
ibm05	8158	11055	19064	47%
ibm06	8158	8715	13966	61%
ibm07	8158	14605	14051	51%
ibm08	8158	15895	16142	60%
ibm09	8158	16395	13780	55%
ibm10	8158	27890	30755	53%
ibm11	16350	10925	19234	59%
ibm12	16350	15545	26748	52%
ibm13	16350	12230	19539	59%
ibm14	16350	25475	26370	61%
ibm15	16350	23785	27284	63%
ibm16	16350	34015	42860	59%
ibm17	16283	38895	45686	56%
ibm18	16350	37065	52846	64%

A. Placement Examples With Global Connections Only

One way to study the impact of global connections is to create circuits with global nets only. Our construction procedure takes an integer p, and a float α, $0 \leq \alpha \leq 1$ as inputs. It outputs a netlist N' and a placement solution S, such that N' has p modules and an aspect ratio of α. Each net in N' connects either an entire row or column, as shown in Fig. 3. The number of nets is the total number of rows and columns. There are no pads in these examples. These examples are similar to datapath placement examples, where data flows horizontally through the bit-slice along buses and control signal flows vertically. One solution to such benchmarks has a configuration similar to Fig. 3, whose wirelength is the sum of the length of each row and column, which is obviously an upper bound of the optimal wirelength.

B. Placement Examples With Nonlocal Connections

The second approach is to introduce nonlocal nets into benchmarks which only have local nets in the optimal solution. Compared with local nets, the nonlocal nets usually have a longer wirelength, and are used to mimic the global connections in our study. We need to further introduce some notations here.

Given a netlist N and a placement solution P, let $l_{\max}$ be the wirelength of the longest net in P. Let $W(N,P) = (w_1, \ldots w_n)$ be the *wirelength distribution vector* (*WDV*), where w_i is the number of nets whose wirelength is between $(i-1) * l_{\max}/n$

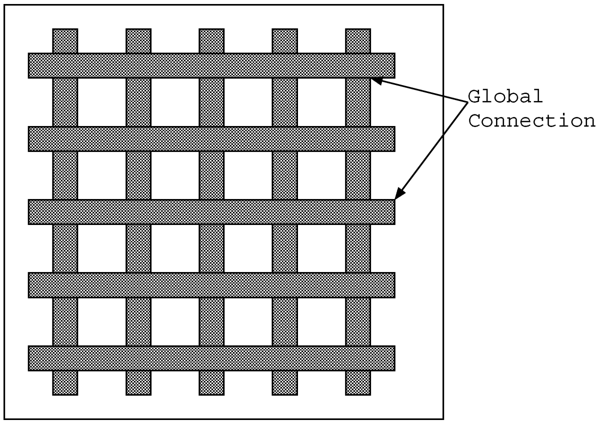

Fig. 3. Circuit with global connections only.

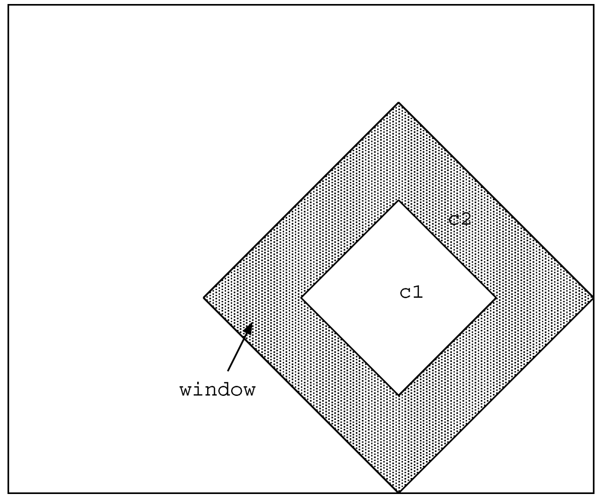

Fig. 4. Nonlocal net generation. One corner of the bounding box is randomly selected and the other is picked within the window satisfying its wirelength range.

and $i*l_{\max}/n$. Without loss of generality, $l_{\max}$ can be given as a percentage of the chip size, and w_i can be given as a percentage of the total number of nets.

We would like to solve the following problem. Given a netlist N, an integer p, two floats $l_{\max}$, α, $0 \leq \alpha \leq 1$, and two vectors $W = (w_1, \ldots w_n)$, $D = (d_1, \ldots d_m)$, construct a new netlist N' and a placement solution S, such that the following is true.

- N' has p modules and has D as its NDV.
- The ratio of nonlocal nets to the total number of nets in S is α.
- The percentage of nonlocal nets with wirelength between $(i-1)*l_{\max}/n$ and $i*l_{\max}/n$ is $\alpha \times w_i / \sum_{j=2}^{n} w_j$, for $i = 2, \ldots, n$.

We extended the algorithm presented in the previous section by relaxing the optimality requirement for a subset of the nets. The new algorithm works in two phases. In the preparation phase, the nodes are put in a $\lceil\sqrt{p}\rceil \times \lceil p/\lceil\sqrt{p}\rceil\rceil$ shaped region. The netlist is scanned and a total of $\alpha \times \sum_{i=2}^{m} d_i$ nets are designated as nonlocal nets. For the nonlocal nets of degree k, a total of $d_k \times w_i / \sum_{j=2}^{n} w_j$ of them are assigned a wirelength range of $((i-1)*l_{\max}/n, i*l_{\max}/n)$, for $i = 2, \ldots, n$. In the generation phase, local nets are generated by the same method as in PEKO. For each nonlocal net, one corner of its bounding box is randomly picked from the chip. The other corner is selected within the window satisfying its wirelength requirement, as shown in Fig. 4. The rest of the cells in the net are randomly picked from sites within the bounding box. In the end, a netlist is extracted from the constructed configuration.

Although the optimal wirelength for the generated circuits is no longer known, we can calculate the bounding-box wirelength of the random nets and add it to the optimal wirelength of the local nets. The sum serves as an upper bound of the optimal wirelength.

IV. Experimental Results and Analysis

A. Experimental Results for the PEKO Suite

We performed our experiments on the PEKO benchmarks with four state-of-the-art placers from academia and one industrial placer.

Dragon: Dragon is based on a multilevel framework. It uses hMetis [36] to derive an initial partitioning result on the circuit, then undergoes a series of refinement stages doing bin-based swapping with simulated annealing [1]. We used Dragon v.2.20, downloaded from [37].

Capo: Capo is built on a multilevel partitioner. It aims to enhance the routability with such techniques as tolerance computation and block splitting heuristics [2]. We used Capo v.8.6 downloaded from [38].

mPL: mPL is also based on a multilevel framework. It uses nonlinear programming to handle the nonoverlapping constraints on the coarsest level, then uses Goto-based [39] relaxation in subsequent refinement stages [3]. We used mPL v.3.0 in our experiment. Compared with [3], mPL v.3.0 uses an additional V cycle and does distance-based clustering in the second V cycle [40]. Further, it uses AMG-based interpolation to derive the starting solution on all the levels except the coarsest level.

mPG: mPG is built on a multilevel framework. It performs first choice (FC) clustering and uses hierarchical density control to minimize the overflow of each placement bin during the refinement process. If necessary, it builds an incremental A-tree to optimize the routability. We used mPG v.1.0 given in [4].

QPlace: QPlace [5] is the placement engine used in the Silicon Ensemble of Cadence. The version we used is QPlace v.5.1, subversion 5.1.55, in Silicon Ensemble v.5.3.

Experiments with Dragon, mPG and QPlace are performed on a SUN Blade 1000 750 Hz running SunOs 5.8 with 4 GB of memory.[4] The experiments with Capo and mPL are performed on a Pentium IV 2.40 GHz running RedHad 8.0 with 2 GB of memory. Since all of the tools make use of randomization, running them multiple times may give different results. The result is the average of five runs. Also, direct comparison of Capo and mPL's runtime with the others may not be meaningful.[5] We need to emphasize that it is not our purpose to give a comparison of

[4]When running QPlace, we set its congestion optimization parameter to "false."

[5]In the tables, we scaled Capo and mPL's runtime according to the CINT2000 value obtained from [41]. However, this may still not be a fair comparison with the other tools.

TABLE V
STATISTICS OF PIN DEGREE AND CUT SIZE FOR PEKO SUITE-1

Circuit	Avg. pin deg	Stdev. pin deg	Avg. V cutsize	Stdev. V cutsize	Avg. H cutsize	Stdev. H cutsize
Peko01	4.004	0.8288	114	4.66667	112.2	7.49518
Peko02	4.17129	0.838969	142	6.8313	143.3	9.03143
Peko03	4.06979	0.791929	156.1	7.40045	153.8	6.0148
Peko04	3.86793	0.804535	162.4	6.32807	161.9	7.68042
Peko05	4.42116	0.784998	176.9	6.24411	177	10.6979
Peko06	3.95429	0.813887	176.7	9.26223	181.7	10.3285
Peko07	3.83586	0.819579	209.3	9.26223	212.8	14.2423
Peko08	4.00443	0.820332	220.6	8.66923	216.6	12.4829
Peko09	4.17093	0.830584	243.5	7.19954	247.5	9.39563
Peko10	4.31068	0.793858	282.4	5.87272	283.4	13.0996
Peko11	3.99096	0.836835	281.8	7.81452	273.8	16.6853
Peko12	4.49305	0.75743	296.5	11.138	294.7	10.2095
Peko13	4.25396	0.819335	315.8	7.89937	317	9.48683
Peko14	3.71058	0.791824	371	13.4495	366.5	8.50163
Peko15	4.4362	0.781188	451.3	8.47283	442.9	7.21803
Peko16	4.25082	0.821527	449.5	14.9462	446.8	8.16224
Peko17	4.64704	0.824663	495.2	7.99722	485.1	9.39799
Peko18	3.8944	0.85021	450.2	21.7092	451.6	15.3275

TABLE VI
EXPERIMENTAL RESULTS FOR SUITE-1

Circuit	Dragon 2.20		Capo 8.6		mPG 1.0		mPL 3.0		QPlace 5.1	
	WR	Runtime(s)	WR	Runtime(s)	WR	Runtime(s)	WR	Runtime(s)	WR	Runtime(s)
Peko01	1.96	1035	1.79	88	1.87	603	1.36	223	1.84	84
Peko02	1.95	1759	1.78	146	2.02	1008	1.45	362	1.82	123
Peko03	2.02	1944	1.80	183	2.10	1222	1.42	433	1.85	156
Peko04	2.29	3669	1.77	229	1.88	1360	1.36	439	1.94	190
Peko05	2.20	5889	1.79	262	1.88	1546	1.39	627	1.72	273
Peko06	2.05	4948	1.81	277	1.96	1759	1.38	593	1.77	267
Peko07	2.22	3500	1.84	415	1.96	3209	1.42	871	1.82	373
Peko08	2.10	9025	1.87	499	1.96	3480	1.44	1115	1.82	481
Peko09	2.10	7373	1.86	555	1.97	5520	1.41	982	1.84	463
Peko10	1.82	9845	1.91	742	1.88	5550	1.45	1476	1.78	660
Peko11	1.94	7803	1.87	760	1.92	4495	1.42	1275	1.88	655
Peko12	1.91	10655	1.85	807	1.90	5528	1.39	1468	1.87	661
Peko13	2.20	10188	1.88	978	1.94	5731	1.40	1653	1.92	694
Peko14	2.00	13168	1.91	1983	2.03	10769	1.53	2765	1.84	1192
Peko15	2.22	18145	1.91	2508	2.10	13898	1.43	3554	1.83	1881
Peko16	2.33	20027	2.03	3061	1.84	8205	1.59	4174	1.75	2037
Peko17	2.38	40292	1.94	3155	1.90	11973	1.50	5082	1.91	2177
Peko18	2.40	36712	1.90	3699	1.97	9304	1.46	5518	1.80	2134
Avg.	2.12		1.86		1.95		1.43		1.83	

the five placers. The experiments are performed to determine how much room is left for improvement in existing placement algorithms.

Table V shows some interesting statistics for the circuits of suite-1. The first columns show the average pin degrees of the modules and their standard deviation. The latter columns show the average cutsizes along the vertical and horizontal cutlines of the chip and the corresponding standard deviations in the optimal placement. Excluded are the vertical cutlines in the white space area. All these values in most of the cases stay in a limited range, a fact that shows the robustness of the benchmarks.

The test results for suite-1 are given in Table VI. For each benchmark, the WR is calculated for the four tools and given in the columns labeled "WR." According to the experiments, none of these tools achieve a WR close to 1. The wirelengths produced by these tools can be 1.59 to 2.40 times the optimal in the worst cases.

It should be noted that there is some difference of white-space utilization between the placers. Fig. 5 gives the placement results produced by them on Peko01. mPL, mPG, and Dragon in wirelength driven mode displays the behavior of variable die placers by packing the cells on each row to the left. Capo and QPlace tend to spread cells across the entire core region, aiming to enhance routability. This will certainly sacrifice the wirelength to some extent. However, given the gap between their wirelengths and the optimal value, there remains significant room for improvement in existing placement algorithms.

The entire test is repeated on suite-2 to observe how the WRs change as the design size grows. Since the benchmark sizes are $10\times$ larger in this set, we set an upper limit of 24 h to a tool's runtime. The results are given in Table VII. QPlace scales well in terms of runtime. It finishes 16 out of 18 benchmarks (up to 1.83 M placeable modules), and runs out of memory on the remaining two (with 1.85 and 2.15 M

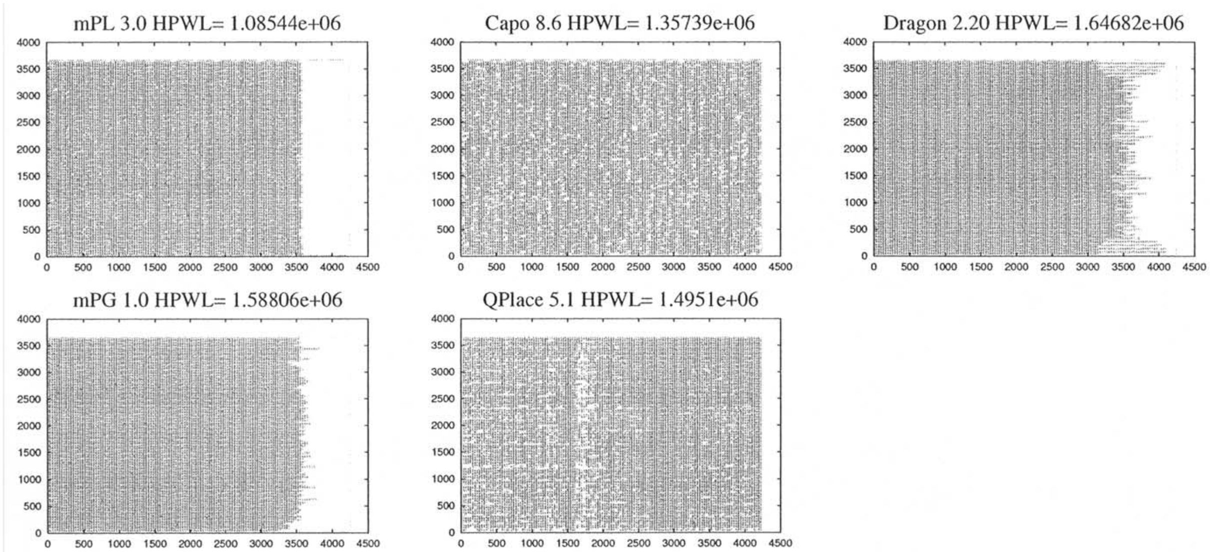

Fig. 5. Placement Results on Peko01. mPL, mPG, and Dragon in wirelength-driven mode pack cells on each row to the left. Capo and QPlace tend to spread cells across the entire core region.

TABLE VII
EXPERIMENTAL RESULTS FOR SUITE-2

Circuit	Dragon 2.20		Capo 8.6		mPG 1.0		mPL 3.0		QPlace 5.1	
	WR	Runtime(s)	WR	Runtime(s)	WR	Runtime(s)	WR	Runtime(s)	WR	Runtime(s)
Peko01x10	2.36	27279	1.91	1555	1.82	8452	1.44	2942	1.88	1221
Peko02x10	2.54	33715	1.96	3071	1.98	13735	1.65	4296	1.86	2084
Peko03x10	2.33	35064	1.97	4240	1.91	7218	1.52	5027	2.07	2216
Peko04x10	1.99	36978	1.92	5627	1.85	13261	1.48	5862	1.96	2469
Peko05x10	2.05	56023	1.95	6022	1.92	10526	1.48	8725	1.92	2958
Peko06x10	2.50	49426	1.95	7160	2.09	10509	1.50	8794	2.00	3059
Peko07x10	NA	$> 24h$	1.97	12223	2.08	25684	1.72	13752	1.91	4752
Peko08x10	NA	$> 24h$	1.99	14532	2.04	26276	NA	out of mem	1.85	5836
Peko09x10	NA	$> 24h$	1.98	16702	2.13	15517	NA	out of mem	1.82	6072
Peko10x10	NA	$> 24h$	2.03	25388	2.19	29548	NA	out of mem	1.87	8052
Peko11x10	NA	$> 24h$	1.98	26713	2.10	20267	NA	out of mem	1.93	7986
Peko12x10	NA	$> 24h$	2.09	27507	2.22	31094	NA	out of mem	1.83	9164
Peko13x10	NA	$> 24h$	2.08	38310	2.22	35226	NA	out of mem	1.93	9825
Peko14x10	NA	$> 24h$	2.04	90684	NA	out of mem	NA	out of mem	2.05	16319
Peko15x10	NA	$> 24h$	NA	out of mem	NA	out of mem	NA	out of mem	2.15	20871
Peko16x10	NA	$> 24h$	NA	out of mem	NA	out of mem	NA	out of mem	1.92	27231
Peko17x10	NA	$> 24h$	NA	out of mem	NA	out of mem	NA	out of mem	NA	out of mem
Peko18x10	NA	$> 24h$	NA	out of mem	NA	out of mem	NA	out of mem	NA	out of mem
Avg.	2.29		1.99		2.04		1.54		1.94	

placeable modules) on our machine's configuration. Its average WR increases by 11% from 1.83 to 1.94. Capo also shows good scalability in runtime. It finishes 14 of the circuits (up to 1.47 M placeable modules) and runs out of memory on the remaining four circuits. Its average WR shows an increase of 13% with the increase in design size. mPL finishes 7 of the 18 benchmarks, and runs out of memory on the remaining circuits. Its average WR increases by 11% from 1.43 to 1.54. Dragon manages to complete the placement for only the first 6 benchmarks (up to 323 k placeable modules) within 24 h. Its average WR increases from 2.12 to 2.29. mPG can place 13 of the circuits, and its average WR increases from 1.95 to 2.04. Figs. 6 and 7 give the combined results for suite-1 and suite-2. They show how the solution quality and runtime of each tool change with the increase in cell numbers.

Tables VIII and IX give the experimental results for suite-3 and suite-4, which have nets connected with pads. For the circuits of suite-3, the wirelengths produced by the placers are 1.54 to 2.53 times the optimal in the worst cases, and are 1.45 to 2.10 times the optimal on the average. Their average solution quality shows deterioration by an additional 6% to 30% when the problem size increases by a factor of ten.

It can be seen from Tables VIII and IX that having nets connected with pads provides some hint about the optimal solution to some placers, especially mPG that shows a 12% improvement, and QPlace that shows a 9% improvement. This is understandable, since in suite-3 and suite-4, modules connected with pads are placed next to the pads in the optimal solutions. Interestingly enough, Dragon, Capo, and mPL do not seem to benefit from the additional information.

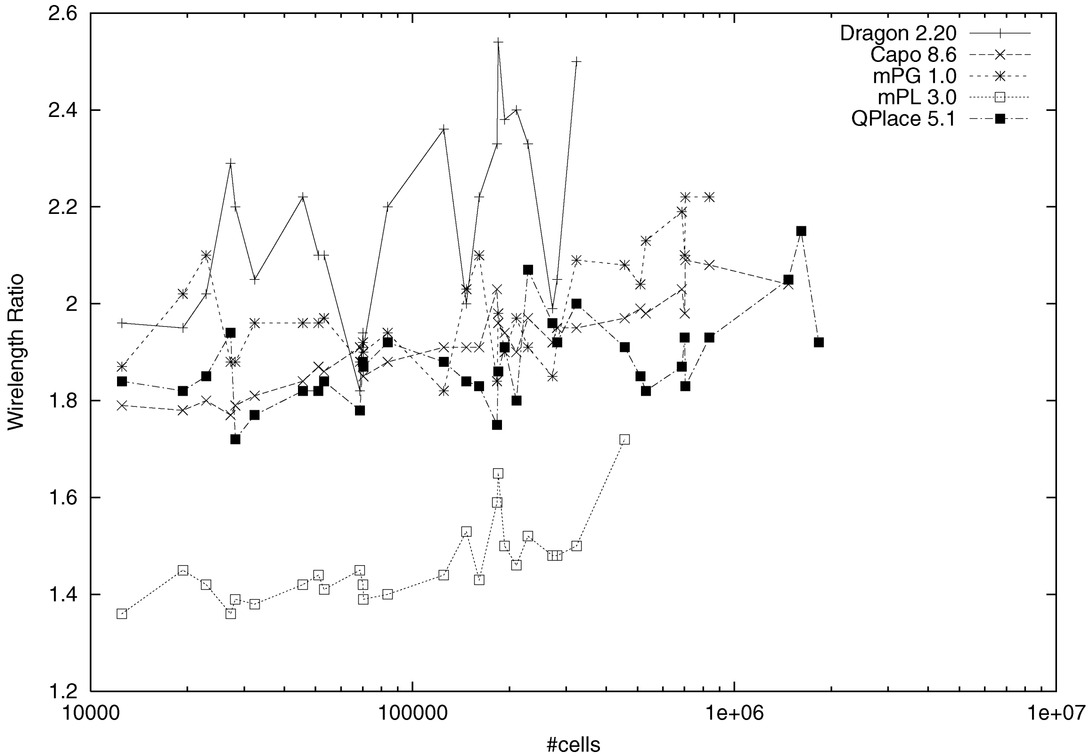

Fig. 6. Solution quality versus cell number (combining suite-1 and suite-2).

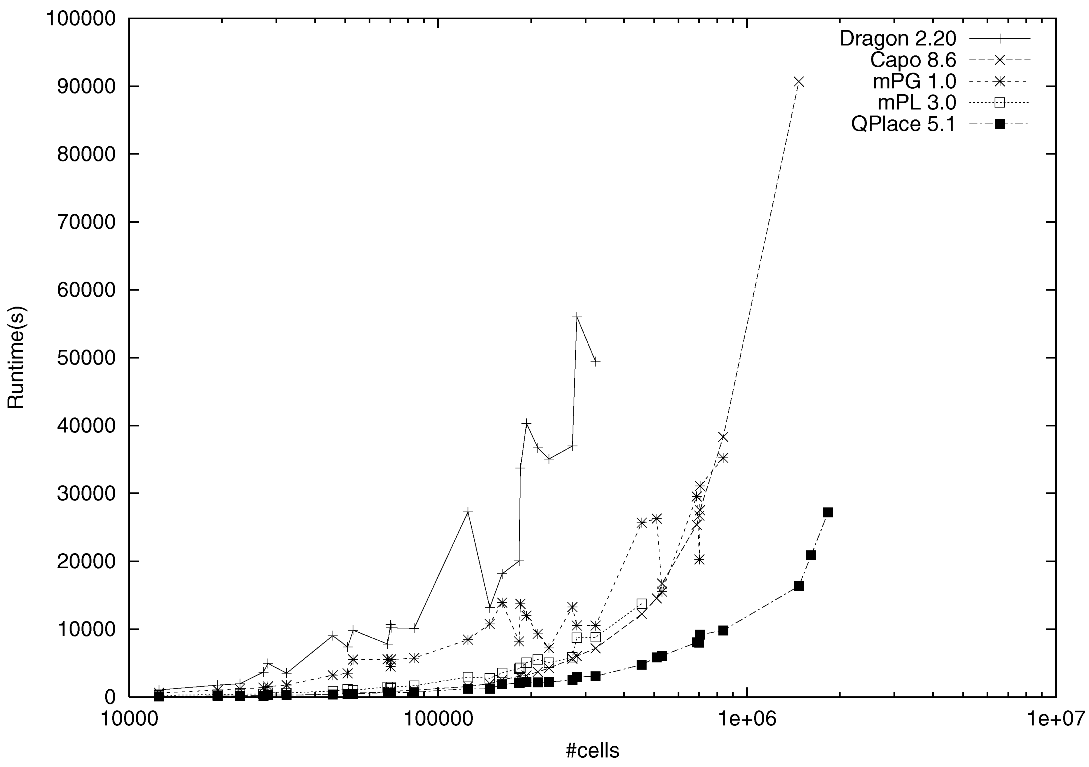

Fig. 7. Runtime versus cell number (combining suite-1 and suite-2).

Although our algorithm is capable of generating arbitrarily-sized benchmarks with known optimal wirelengths, given the scalability problems encountered by these tools on suite-2 and suite-4, it is not meaningful to construct larger designs to further evaluate these algorithms.

B. Experimental Results for Benchmarks With Nonlocal Nets

Using the module numbers extracted from ISPD'98 and an aspect ratio of 1, we generated a set of circuits with global nets only. The circuits are named GPeku01 to GPeku18 and are grouped as the G-PEKU suite (Global nets only Placement

TABLE VIII
EXPERIMENTAL RESULTS FOR SUITE-3

Circuit	Dragon 2.20		Capo 8.6		mPG 1.0		mPL 3.0		QPlace 5.1	
	WR	Runtime(s)	WR	Runtime(s)	WR	Runtime(s)	WR	Runtime(s)	WR	Runtime(s)
Peko01	2.01	1044	1.79	100	1.73	535	1.41	246	1.74	98
Peko02	1.94	1773	1.81	171	2.10	884	1.45	398	1.87	127
Peko03	1.99	1951	1.83	207	1.97	993	1.45	466	1.77	162
Peko04	2.31	3726	1.81	250	1.53	967	1.40	508	1.76	196
Peko05	2.15	7951	1.90	269	1.59	900	1.51	703	1.74	197
Peko06	1.95	7536	1.83	305	1.92	1690	1.40	646	1.71	242
Peko07	2.23	3364	1.89	460	1.71	1805	1.44	995	1.71	373
Peko08	2.05	8615	1.84	524	1.88	2998	1.41	1235	1.71	491
Peko09	2.18	7073	1.84	588	1.98	2912	1.48	1097	1.67	495
Peko10	1.87	9424	1.90	820	1.63	3458	1.54	1623	1.76	569
Peko11	1.98	7555	1.85	757	1.73	1697	1.43	1386	1.75	596
Peko12	1.72	10294	1.88	778	1.88	2480	1.43	1658	1.77	659
Peko13	1.86	9869	1.88	1008	1.80	4286	1.44	1768	1.80	701
Peko14	2.05	12603	1.90	1943	2.00	8276	1.51	3119	1.70	1399
Peko15	2.23	17163	1.87	2509	2.27	16840	1.43	3887	1.70	1752
Peko16	2.29	18852	1.95	2993	1.67	5004	1.46	4759	1.78	1880
Peko17	2.46	38578	1.90	3156	1.69	6174	1.46	5464	1.68	2047
Peko18	2.53	35122	1.93	3638	1.89	7100	1.46	5873	1.71	2069
Avg.	2.10		1.87		1.83		1.45		1.74	

TABLE IX
EXPERIMENTAL RESULTS FOR SUITE-4

Circuit	Dragon 2.20		Capo 8.6		mPG 1.0		mPL 3.0		QPlace 5.1	
	WR	Runtime(s)	WR	Runtime(s)	WR	Runtime(s)	WR	Runtime(s)	WR	Runtime(s)
Peko01x10	2.25	24149	1.94	1568	2.04	3860	1.57	2470	1.74	998
Peko02x10	2.49	33959	1.98	3089	2.10	7758	1.61	3967	1.79	1742
Peko03x10	2.49	34191	2.06	4252	2.06	10037	1.64	4738	1.90	1838
Peko04x10	2.02	36569	1.97	5678	1.92	11636	1.55	5332	1.82	2110
Peko05x10	2.14	57099	2.06	6135	2.19	16514	1.85	7020	1.73	2543
Peko06x10	2.48	52540	1.95	7135	1.96	13138	NA	Abort	1.94	3006
Peko07x10	NA	$> 24h$	2.03	12369	2.09	20846	1.64	14121	1.71	4082
Peko08x10	NA	$> 24h$	1.99	14672	2.59	47765	NA	out of mem	1.72	5316
Peko09x10	NA	$> 24h$	2.03	16964	2.23	26364	NA	out of mem	1.76	5359
Peko10x10	NA	$> 24h$	2.08	25662	NA	out of mem	NA	out of mem	1.74	6756
Peko11x10	NA	$> 24h$	2.01	26402	NA	out of mem	NA	out of mem	1.78	6627
Peko12x10	NA	$> 24h$	2.09	27856	NA	out of mem	NA	out of mem	1.81	7429
Peko13x10	NA	$> 24h$	2.07	39003	NA	out of mem	NA	out of mem	1.81	8236
Peko14x10	NA	$> 24h$	2.12	91484	NA	out of mem	NA	out of mem	1.84	14632
Peko15x10	NA	$> 24h$	NA	out of mem	NA	out of mem	NA	out of mem	1.82	21937
Peko16x10	NA	$> 24h$	NA	out of mem	NA	out of mem	NA	out of mem	1.83	25616
Peko17x10	NA	$> 24h$	NA	out of mem	NA	out of mem	NA	out of mem	NA	out of mem
Peko18x10	NA	$> 24h$	NA	out of mem	NA	out of mem	NA	out of mem	NA	out of mem
Avg.	2.31		2.02		2.13		1.64		1.80	

Examples with Known Upper bound of wirelength). We also generated several sets of benchmarks with nonlocal nets. We call these benchmarks the PEKU suite (Placement Examples with Known Upper bound of wirelength). The parameter α is gradually increased from 0.25% to 10%. The module number and NDVs are derived from ISPD'98. To get the wirelength distribution of nonlocal nets for each circuit, we extracted the WDVs from ISPD circuits placed by Dragon. For each w_i in the WDV, we multiply it by a randomly generated coefficient β, $0.5 \leq \beta \leq 1.5$, so that the created examples are not biased for Dragon. Circuits in PEKU do not have nets connected with pads. The G-PEKU and PEKU circuits used in our study can be downloaded from [42].

We use the same five placers as in the previous section. First, we tested the five placers on five circuits in G-PEKU. The experimental results are given in Table X. The results are the average of five runs for each placer. The WR is now calculated as the ratio of a placement's wirelength to the upper bound of wirelength. Among the five placers, Capo gives the closest solution to the upper bounds.[6] For these examples

TABLE X
EVALUATION RESULTS ON G-PEKU

Circuit	Dragon 2.20 WR	Capo 8.6 WR	mPG 1.0 WR	mPL 3.0 WR	QPlace 5.1 WR
GPeku01	1.98	1.54	1.91	NA	1.94
GPeku05	2.01	1.68	1.97	NA	1.97
GPeku10	2.02	1.72	1.98	NA	1.99
GPeku15	1.99	1.76	1.97	NA	1.99
GPeku18	2.02	1.74	1.98	NA	2.00
Avg.	2.01	1.69	1.96	NA	1.98

[6]Since mPL prunes all the nets with a degree higher than 60, it gives no results on G-PEKU.

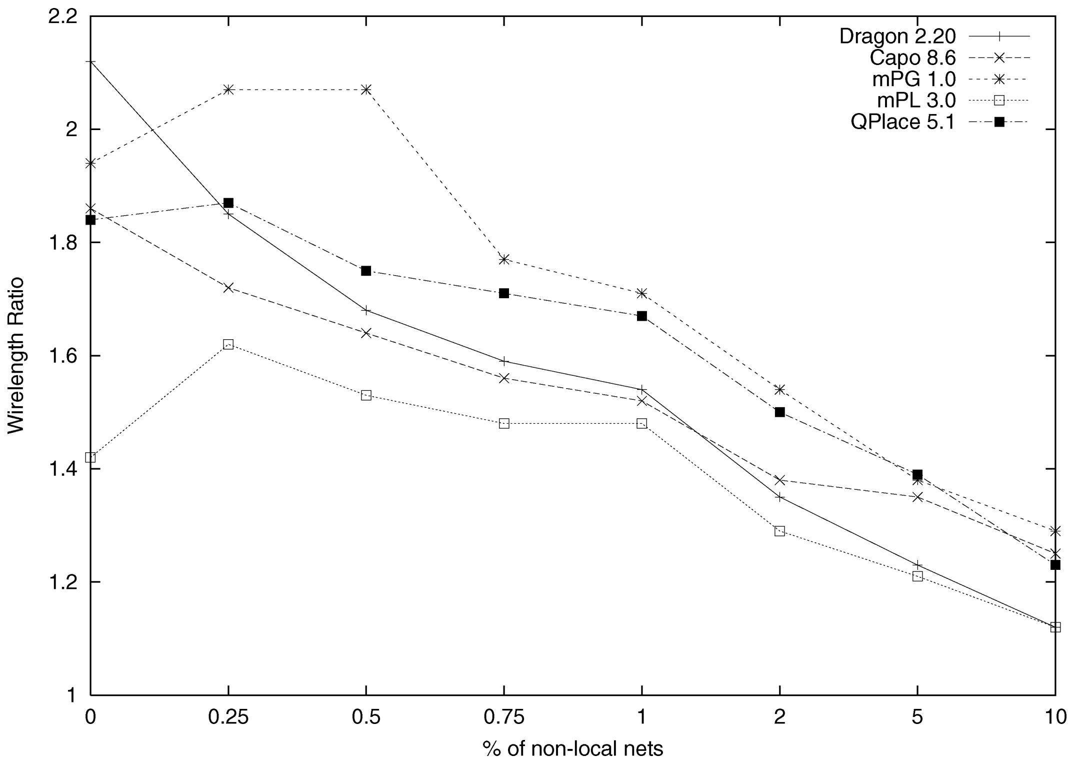

Fig. 8. WR versus percentage of nonlocal nets.

with global nets only, the gap between their solutions and the upper bound varies between 69% and 101% on average, which are similar to the results obtained on PEKO, which has local nets only. This is another validation that there is significant room for improvement for the placement problem.

We also tested the placers on the PEKU benchmarks. For each α, we picked five of the circuits and fed them into the placers. Each circuit was placed three times by the placers. Table X and Fig. 8 show the experimental results for a subset of the PEKU examples, as the value of α changes from 0 up to 0.1 (for $\alpha = 0$, the examples are actually from the PEKO suite). The first column gives the ratio of nonlocal nets to the total nets. Column "LB" gives the lower bound of the optimal wirelength, assuming that each net can be optimally placed. The upper bound of each circuit is given in column "UB." The last five columns give the WR of the five placers. It can be observed that the WRs are decreasing with the increase of nonlocal nets. However, this does not necessarily indicate that the solution quality of the placers is improving. We believe that this is due to the upper bounds of wirelength becoming looser as the percentage of nonlocal nets increases. Therefore, the absolute value of the WRs may not be meaningful. However, comparing WRs from different placers can help us identify the technique that works best under each scenario. Also, comparing the WRs of the same placer can test a placer's sensitivity to global connections. It can be seen that the relative ranking of the placers changes as the percentage of global nets increases.

Combining the results from Tables X and XI, we can make the following observations.

i) None of the placers performs consistently better than the others. Without global nets, mPL gives the shortest wirelength. However, the effectiveness of Dragon improves dramatically with the increase of nonlocal nets. When the percentage of nonlocal nets reaches 10%, it gives the shortest wirelength among the five placers. For examples with global connections only, Capo gives the closest solutions to the upper bounds. The effectiveness of a placer can vary significantly for designs of similar sizes but different characteristics.

ii) The study suggests that new hybrid techniques, which are more scalable and stable, may be needed for future generations of placement tools.

V. Conclusion and Future Work

In this paper, we implemented an algorithm for generating synthetic benchmarks that have known optimal wirelengths and can match any given net distribution vector. Using benchmarks of 10 k to 2 M placeable modules with known optimal solutions, we experimented with four state-of-the-art placers from academia, Dragon [1], Capo [2], mPL [3], and mPG [4], and a leading edge industrial placer, QPlace [5] from Cadence. For the first time, our study reveals the gap between the results produced by these tools versus true optimal solutions. The wirelengths produced by these tools are 1.59 to 2.40 times the optimal in the worst cases, and are 1.43 to 2.12 times the optimal on the average. As for scalability, the average solution quality of each tool deteriorates by an additional 9% to 17% when the problem size increases by a factor of ten. We also studied the impact of nonlocal nets on the quality of the placers by extending the PEKO algorithm to generate synthetic placement benchmarks with a known upper bound of the optimal wirelength. Even for these benchmarks, the wirelengths produced by these tools are 1.75 to 2.18 times the

TABLE XI
EXPERIMENTAL RESULTS FOR THE PEKU CIRCUITS

% of non-local nets	Circuit	LB	UB	Dragon 2.20 WR	Capo 8.6 WR	mPG 1.0 WR	mPL 3.0 WR	QPlace 5.1 WR
0	Peko01	8.14E+05	8.14E+05	1.96	1.79	1.87	1.36	1.69
	Peko05	1.91E+06	1.91E+06	2.20	1.79	1.88	1.39	1.95
	Peko10	4.73E+06	4.73E+06	1.82	1.91	1.88	1.45	1.88
	Peko15	1.15E+07	1.15E+07	2.22	1.91	2.10	1.43	1.85
	Peko18	1.32E+07	1.32E+07	2.40	1.90	1.97	1.46	1.84
	Avg.			2.12	1.86	1.94	1.42	1.84
0.25%	Peku01	8.14E+05	9.24E+05	1.87	1.75	2.04	1.43	1.83
	Peku05	1.91E+06	2.48E+06	1.93	1.71	1.84	1.49	1.79
	Peku10	4.73E+06	6.44E+06	1.67	1.70	2.10	1.75	1.86
	Peku15	1.15E+07	1.69E+07	1.84	1.71	2.17	1.76	1.90
	Peku18	1.32E+07	2.03E+07	1.94	1.74	2.18	1.69	1.97
	Avg.			1.85	1.72	2.07	1.62	1.87
0.50%	Peku01	8.14E+05	1.01E+06	1.73	1.66	1.97	1.39	1.72
	Peku05	1.91E+06	2.93E+06	1.69	1.64	1.72	1.35	1.60
	Peku10	4.73E+06	7.64E+06	1.53	1.68	2.62	1.63	1.75
	Peku15	1.15E+07	2.14E+07	1.67	1.62	2.03	1.81	1.98
	Peku18	1.32E+07	2.50E+07	1.77	1.57	1.98	1.49	1.73
	Avg.			1.68	1.64	2.07	1.53	1.75
0.75%	Peku01	8.14E+05	1.08E+06	1.70	1.62	1.81	1.30	1.77
	Peku05	1.91E+06	3.34E+06	1.65	1.55	1.61	1.30	1.61
	Peku10	4.73E+06	8.67E+06	1.47	1.58	1.87	1.82	1.75
	Peku15	1.15E+07	2.54E+07	1.56	1.57	1.82	1.64	1.84
	Peku18	1.32E+07	2.90E+07	1.59	1.49	1.76	1.36	1.57
	Avg.			1.59	1.56	1.77	1.48	1.71
1%	Peku01	8.14E+05	1.16E+06	1.65	1.60	1.67	1.33	1.76
	Peku05	1.91E+06	3.70E+06	1.59	1.54	1.66	1.45	1.60
	Peku10	4.73E+06	9.59E+06	1.43	1.56	1.77	1.63	1.75
	Peku15	1.15E+07	2.91E+07	1.51	1.51	1.72	1.53	1.64
	Peku18	1.32E+07	3.25E+07	1.53	1.40	1.73	1.46	1.62
	Avg.			1.54	1.52	1.71	1.48	1.67
2%	Peku01	8.14E+05	1.41E+06	1.47	1.47	1.64	1.30	1.59
	Peku05	1.91E+06	5.04E+06	1.36	1.34	1.36	1.22	1.38
	Peku10	4.73E+06	1.24E+07	1.30	1.37	1.58	1.41	1.56
	Peku15	1.15E+07	4.09E+07	1.28	1.38	1.46	1.30	1.44
	Peku18	1.32E+07	4.33E+07	1.36	1.34	1.64	1.21	1.55
	Avg.			1.35	1.38	1.54	1.29	1.50
5%	Peku01	8.14E+05	1.95E+06	1.29	1.37	1.38	1.18	1.35
	Peku05	1.91E+06	8.13E+06	1.18	1.25	1.37	1.22	1.22
	Peku10	4.73E+06	1.78E+07	1.20	1.29	1.38	1.19	1.46
	Peku15	1.15E+07	6.35E+07	1.21	1.39	1.40	1.18	1.46
	Peku18	1.32E+07	6.38E+07	1.25	1.45	1.39	1.25	1.44
	Avg.			1.23	1.35	1.38	1.21	1.39
10%	Peku01	8.14E+05	2.57E+06	1.17	1.21	1.23	1.07	1.30
	Peku05	1.91E+06	1.20E+07	1.10	1.18	1.51	1.17	1.23
	Peku10	4.73E+06	2.36E+07	1.10	1.25	1.23	1.08	1.27
	Peku15	1.15E+07	8.45E+07	1.10	1.23	1.21	1.07	1.19
	Peku18	1.32E+07	8.65E+07	1.11	1.38	1.28	1.19	1.17
	Avg.			1.12	1.25	1.29	1.12	1.23

wirelength upper bound in the worst case, and are 1.62 to 2.07 times the wirelength upper bound on the average. Moreover, none of the placers produces consistently better results than the others with the presence of global nets. The fact that there is 50% to 150% room for placement quality improvement is significant. If this quality gap could be closed, the resulting benefit would be equivalent to advancing several technology generations. In comparison, the introduction of copper interconnects is equivalent to a 30% wirelength reduction, and so is each process technology scaling. But each of these requires multibillion dollar investments.

Our study is by no means complete. We did not have a chance to experiment with a number of well known placers, such as Gordian-L [8], TimberWolf [9] from academia, as well as commercial placement tools from Synopsys, Avanti! etc. Also, the benchmarks generated by our algorithm have several limitations. For example, all modules in these circuits are of uniform size, making them unsuitable for evaluating the legalization capability of detailed placement algorithms. Therefore, obtaining good results for these benchmarks may not guarantee good solution quality in real circuits. Finally, these benchmarks can not be used to evaluate routability and performance. Nevertheless, we have made a very important step in understanding the optimality and scalability of existing placement algorithms. We plan to further enhance our benchmark construction algorithm and broaden its applicability in the future.

ACKNOWLEDGMENT

The authors would like to thank X. Yuan for providing the data of mPG. They would like to thank J. Shinnerl and K. Sze for providing the experimental data of mPL. They would also like to thank Prof. I. Markov for providing Capo's latest version for their experiment.

References

[1] M. Wang, X. Yang, and M. Sarrafzadeh, "Dragon2000: Standard-cell placement tool for large industry circuits," in *Proc. Int. Conf. Computer-Aided Design*, 2000, pp. 260–264.

[2] A. E. Caldwell, A. B. Kahng, and I. L. Markov, "Can recursive bisection produce routable placements?," in *Proc. Design Automation Conf.*, 2000, pp. 477–482.

[3] T. F. Chan, J. Cong, T. Kong, and J. R. Shinnerl, "Multilevel optimization for large-scale circuit placement," in *Proc. Int. Conf. Computer-Aided Design*, 2000, pp. 171–176.

[4] C.-C Chang, J. Cong, Z. Pan, and X. Yuan, "Physical hierarchy generation with routing congestion control," in *Proc. Int. Symp. Physical Design*, 2002, pp. 36–41.

[5] "Envisia ultra placer reference," in *QPlace Version 5.1.55*: Cadence Design Systems Inc., 1999.

[6] *International Technology Roadmap for Semiconductors 2001 Edition*, Semiconductor Industry Association, 2003.

[7] J. Cong, "An interconnect-centric design flow for nanometer technologies," *Proc. IEEE*, vol. 89, pp. 505–528, Apr. 2001.

[8] G. Sigl, K. Doll, and F. Johannes, "Analytical placement: A linear or quadratic objective function," in *Proc. Design Automation Conf.*, 1991, pp. 427–432.

[9] C. Sechen and A. Sangiovanni-Vincentelli, "The TimberWolf placement and routing package," *IEEE J. Solid-State Circuits*, vol. 20, pp. 432–439, Feb. 1985.

[10] S. N. Adya, M. Yildiz, I. L. Markov, P. G. Villarrubia, P. N. Parakh, and P. H. Madden, "Benchmarking for large-scale placement and beyond," in *Proc. Int. Symp. Physical Design*, 2003, pp. 95–103.

[11] P. Madden, "Reporting of standard cell placement results," *IEEE Trans. Computer-Aided Design*, vol. 21, pp. 240–247, Feb. 2002.

[12] J. M. Kleinhans, G. Sigl, and F. M. Johannes, "Gordian: A new global optimization/rectangle dissection method for cell placement," in *Proc. Int. Conf. Computer-Aided Design*, 1988, pp. 506–509.

[13] W. Sun and C. Sechen, "Efficient and effective placement for very large circuits," in *Proc. Int. Conf. Computer-Aided Design*, 1993, pp. 170–177.

[14] H. Eisenmann and F. M. Johannes, "Generic global placement and floorplanning," in *Proc. Design Automation Conf.*, 1998, pp. 269–274.

[15] S. Hur and J. Lillis, "Mongrel: Hybrid techniques for standard cell placement," in *Proc. Int. Conf. Computer-Aided Design*, 2000, pp. 165–170.

[16] X. Yang, B. Choi, and M. Sarrafzadeh, "Routability driven white space allocation for fixed-die standard-cell placement," in *Proc. Int. Symp. Physical Design*, 2002, pp. 42–49.

[17] X. Yuan, personal communication, 2002.

[18] J. Lillis, personal communication, 2001.

[19] C. J. Alpert, "The ISPD'98 circuit benchmark suite," in *Proc. Int. Symp. Physical Design*, 1998, pp. 85–90.

[20] D. Ghosh, N. Kapur, and F. Brglez, "Toward a new benchmarking paradigm in EDA: Analysis of equivalent class mutant circuit distribution," in *Proc. Int. Symp. Physical Design*, 1997, pp. 136–144.

[21] K. Iwama and K. Hino, "Random generation of test instance for logic optimizers," in *Proc. Design Automation Conf.*, 1994, pp. 430–434.

[22] J. Darnauer and W. Dai, "A method for generating random circuits and its application to routability measurement," in *Proc. Int. Symp. Field Program. Gate Arrays*, 1996, pp. 66–72.

[23] D. Stroobandt, P. Verplaetse, and J. V. Campenhout, "Generating synthetic benchmark circuits for evaluating CAD tools," *IEEE Trans. Computer-Aided Design*, vol. 19, pp. 1011–1022, Sept. 2000.

[24] ——, "Toward synthetic benchmark circuits for evaluating timing-driven CAD tools," in *Proc. Int. Symp. Physical Design*, 1999, pp. 60–66.

[25] M. Hutton, J. Rose, J. Grossman, and D. Corneil, "Characterization and parameterized generation of synthetic combinatinational circuits," *IEEE Trans. Computer-Aided Design*, vol. 17, pp. 985–996, Oct. 1998.

[26] B. S. Landman and R. L. Russo, "On a pin versus block relationship for partitions of logic graphs," *IEEE Trans. Computers*, vol. 20, pp. 1469–1479, Dec. 1971.

[27] G. Parthasarthy, M. Marek-Sadowska, A. Mukherjee, and A. Singh, "Interconnect complexity-aware FPGA placement using Rent's rule," in *Proc. System-Level Interconnect Prediction*, 2001, pp. 115–121.

[28] L. W. Hagen, D. J.-H. Huang, and A. B. Kahng, "Quantified suboptimality of VLSI layout heuristics," in *Proc. Design Automation Conf.*, 1995, pp. 216–221.

[29] W. J. Dally and A. Chang, "The role of custom design in ASIC chips," in *Proc. Design Automation Conf.*, 2000, pp. 643–647.

[30] K. Boese, personal communication, 2002.

[31] C. C. Chang, J. Cong, and M. Xie, "Optimality and scalability study of existing placement algorithms," in *Proc. Asia South Pacific Design Automation Conf.*, 2003, pp. 621–627.

[32] J. Cong, M. Romesis, and M. Xie, "Optimality, scalability and stability study of partitioning and placement algorithms," in *Proc. Int. Symp. Physical Design*, 2003, pp. 88–94.

[33] [Online]. Available: http://www.eedesign.com/story/OEG20030205S-0014.

[34] [Online]. Available: http://www.eedesign.com/story/OEG20030410S-0029.

[35] [Online]. Available: http://cadlab.cs.ucla.edu/~pubbench/peko.htm.

[36] G. Karypis, B. Aggarwal, V. Kumar, and S. Shekhar, "Multi-level hypergraph partitioning: Application in VLSI domain," *IEEE Trans. VLSI Syst.*, vol. 7, pp. 69–79, 1999.

[37] [Online]. Available: http://er.cs.ucla.edu/Dragon/download.html.

[38] [Online]. Available: http://vlsicad.ucsd.edu/GSRC/bookshelf/Slots/Placement/bin/.

[39] S. Goto, "An efficient algorithm for the two-dimensional placement problem in electrical circuit layout," *IEEE Trans. Circuit Systems*, vol. CAS-28, pp. 12–18, Jan. 1981.

[40] N. K. Sze, personal communication, 2002.

[41] [Online]. Available: http://www.specbench.org/cgi-bin/osgresults.

[42] [Online]. Available: http://cadlab.cs.ucla.edu/~pubbench/peku.htm.

Chin-Chih Chang received the B.S. degree from National Taiwan University, Taipei, Taiwan, R.O.C., in 1989, the M.S. degree from the State University of New York, Stony Brook, in 1993, and the Ph.D. degree from the University of California, Los Angeles, in 2002, all in computer science.

He joined Cadence Design Systems, Inc., San Jose, CA, in 2002. His research interests include VLSI CAD algorithms on placement and routing.

Jason Cong (S'88–M'90–SM'96–F'00) received the B.S. degree from Peking University, Beijing, China, in 1985, and the M.S. and Ph.D. degrees from the University of Illinois, Urbana-Champaign, in 1987 and 1990, respectively, all in computer science.

Currently, he is a Professor and Co-Director of the VLSI CAD Laboratory in the Computer Science Department, University of California, Los Angeles. He has been appointed as a Guest Professor at Peking University since 2000. He has published over 170 research papers and led over 20 research projects supported by the Defense Advanced Research Projects Agency, the National Science Foundation, the Semiconductor Research Corporation (SRC), and a number of industrial sponsors in these areas. His research interests include layout synthesis and logic synthesis for high-performance low-power VLSI circuits, design and optimization of high-speed VLSI interconnects, field-programmable gate array (FPGA) synthesis, and reconfigurable computing.

Prof. Cong has served as the General Chair of the 1993 ACM/SIGDA Physical Design Workshop, the Program Chair and General Chair of the 1997 and 1998 International Symposia on FPGAs, respectively, and on program committees of many VLSI CAD conferences, including the Design Automation Conference, International Conference on Computer-Aided Design, and International Symposium on Circuits and Systems. He is an Associate Editor of *ACM Transactions on Design Automation of Electronic Systems* and IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS. He received the Best Graduate Award from Peking University, in 1985, and the Ross J. Martin Award for Excellence in Research from the University of Illinois, in 1989. He received the Research Initiation and Young Investigator Awards from the National Science Foundation, in 1991 and 1993, respectively. He received the Northrop Outstanding Junior Faculty Research Award from the University of California, in 1993, and the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN Best Paper Award in 1995. He received the ACM Recognition of Service Award in 1997, the ACM Special Interest Group on Design Automation Meritorious Service Award in 1998, and the Inventor Recognition and Technical Excellence Awards from the SRC, in 2000 and 2001, respectively.

Michail Romesis (S'00) received the B.S. degree in electrical and computer engineering from the National Technical University of Athens, Athens, Greece, in 1999 and the M.S. degree in computer science from the University of California, Los Angeles, in 2001. He is currently pursuing the Ph.D. degree in computer science at the University of California, Los Angeles.

His research interests include very large scale integration computer-aided design algorithms for placement and floorplanning.

Mr. Romesis received the Dimitris Chorafas Foundation Award in 2003.

Min Xie received the B.E. degree in computer science from Tongji University, Shanghai, China, in 1997 and the M.S. degree in computer science from Tsinghua University, Beijing, China, in 2001. He is currently pursuing the Ph.D. degree in computer science at the University of California, Los Angeles.

His research interests include very large scale integration physical design, placement, and global routing.