A Study of Netlist Structure and Placement Efficiency

Qinghua Liu
Department of Electrical and Computer Engineering
Univ. of California, Santa Barbara, CA 93106, USA
1-805-893-5678
qinghual@ece.ucsb.edu

Malgorzata Marek-Sadowska
Department of Electrical and Computer Engineering
Univ. of California, Santa Barbara, CA 93106, USA
1-805-893-2721
mms@ece.ucsb.edu

ABSTRACT
In this paper, we study the relationship between netlist structure and the efficiency of placers measured in terms of quality and stability of results. We analyze three types of placers: analytic, simulated-annealing-based and partition-based. We test the placers on industrial and synthetic benchmarks. Based on the observations and analyses of experimental results, we obtain several useful conclusions about relationships between netlist structure and placement efficiency of different types of placers.

Categories and Subject Descriptors
J.6 Computer-Aided Engineering-Computer-aided Design (CAD)

General Terms
Algorithms

Keywords
Netlist Structure, Placement, Efficiency

1. INTRODUCTION
Early logic synthesis, when little physical information is available, has significant impact on the structure of logic circuits. The structure remains unchanged during the placement process. The quality of placement depends not only on the placement algorithms but also on the structure of the logic network. Consequently, physical and logic co-synthesis have attracted a lot of research effort in the recent years. Pedram and Bhat [16] developed a mapper which considered layout area and wire delay during the technology-dependent phase of logic synthesis. The same authors [17] presented techniques to integrate interconnection optimization with logic restructuring and technology decomposition phases of logic synthesis. Kutzschebauch and Stok [15] proposed congestion-aware algorithms for layout-driven decomposition and technology mapping, two of the steps that have the most significant impact on congestion during logic synthesis, to effectively decrease wire length and improve congestion. Kudva and Sullivan [14] observed that *adhesion*, a property of network structure, can be linked to routing congestion. They also provided a metric for adhesion and showed that it can be used, in addition to literal count, to estimate and optimize post-routing congestion early in the design flow. Though some promising results have been shown in those works, we still have not captured the whole picture of the relationship between logic structure and placement efficiency.

In [1], the major publicly available standard-cell placement tools have been analyzed. The authors compared the half-perimeter wire lengths achieved by the state-of-the-art academic placers on different benchmark suites. From the experimental data, the authors observed that different placers usually have different placement efficiency on different benchmark suites. For example, Dragon performs well on IBM-Place benchmarks and mPL performs well on PEKO benchmarks. Then, based on empirical analysis, the authors stated that: (i) it may be difficult for an annealer (Dragon) to place regular structures; (ii) analytic placer (KraftWerk) does not work well on netlists with numerous multi-pin nets. The comparison results in [1] reveal that netlist structure plays different roles for different placers. Netlist structure friendly to one placer may not be friendly to another placer. So to understand the relationship between netlist structure and placement efficiency, we need to consider different placers individually.

In this paper, we study the relationship between netlist structure and placement efficiency in different placement tools. For placement efficiency, we consider both placement quality and placement stability. We analyze three types of placers: analytic, simulated-annealing-based and partition-based. We experiment with industrial and synthetic benchmarks. Based on our observation and analysis of experimental results, we summarize several useful conclusions on the relationship between netlist structure and placement efficiency of different placers.

The rest of this paper is organized as follows. In Section 2 we introduce the placers and benchmarks used in our work. In Section 3 we look into some global characteristics of netlist structure and correlate them with the placement efficiency of different placers. In Section 4 we study one class of special benchmarks, the I-PEKO suite, which contain examples with pre-known optimal results and whose net distribution statistics mimic the IBM benchmarks. We conclude the paper in Section 5.

2. PLACERS AND BENCHMARKS
2.1 Placers
Current placement approaches can be grouped into three major categories: analytic approaches (linear programming and force-directed methods), simulated annealing, and

Table 1. **PEKO suites with fixed net number and varying net degree distribution**

Bench	Capo			Gold			mPL		
	Suite1	Suite2	Suite3	Suite1	Suite2	Suite3	Suite1	Suite2	Suite3
test01	1.406	1.527	1.660	1.307	1.366	1.436	N/A	1.268	1.299
test02	1.445	1.619	1.768	1.329	1.458	1.529	1.331	1.290	1.337
test03	1.404	1.608	1.768	1.298	1.455	1.540	1.359	1.338	1.382
test04	1.438	1.611	1.740	1.318	1.438	1.519	1.332	1.270	1.321
test05	1.433	1.609	1.791	1.314	1.455	1.555	1.421	1.374	1.428
test06	1.450	1.658	1.830	1.341	1.508	1.591	1.383	1.362	1.365
test07	1.442	1.633	1.791	1.328	1.470	1.563	1.465	1.469	1.435
test08	1.444	1.649	1.828	1.326	1.483	1.587	1.435	1.369	1.397
test09	1.451	1.638	1.897	1.335	1.464	1.648	1.413	1.382	1.357
Ave	1.435	1.617	1.786	1.322	1.455	1.552	1.392	1.347	1.369

partitioning-based. In this work, we experiment with four placers, representatives from each category.

Capo [4][5] is a global fixed-die placer based on recursive min-cut bisection. Min-cut hypergraph partitioning of netlists with over 200 cells is performed using the modular implementation [3] of multilevel Fiduccia-Mattheyses (FM) heuristic [2][13]. A *flat* FM heuristic [11] is used independently for instances with 35-200 cells. Smaller instances are solved optimally with the branch-and-bound approach. In all the experiments reported in this paper, we use Capo8.7 [23] version.

Dragon [21][22] performs a recursive min-cut partitioning using hMetis [13]. At each partition level, the sub-circuit inside a global bin is quadrisected into four smaller sub-circuits. The post-bin swapping step minimizes the overall wire length. This process continues until each global bin contains 3-7 cells. Then a simulated annealing-based refinement is applied on the global placement, followed by detailed placement.

mPL [6][7] first coarsens the netlist by performing recursively edge-separability clustering [6] or first-choice clustering [7]. Once the cell cardinality has been reduced to about 1000, a nonlinear program is solved by a customized interior-point method. Slot assignment and discrete refinement are then performed to improve the continuous approximation results. Finally, recursive de-clustering and refinement are applied to map the results back to the fine-grain solution. Quadratic relaxation on noncontagious subsets, together with algebraic multigrid interpolation and multiple V-cycle iterations are used in [7] to improve the solution. We experiment with the mPL2.1 version from [23].

Gold is a simulated annealing-based placer implemented in the FPI [12] framework. It follows the two-stage global and detailed placement flow [18]. We set a very slow annealing schedule during the global placement stage to guarantee good placement results. We refer to the placement results generated by the Gold placer as *Gold solutions*.

2.2 Benchmarks

IBM-Place [21] are industrial benchmarks. Several variants of these benchmarks are available on-line. We use the original suite released in the year 2000. In the following, we refer to this particular suite as the *IBM suite*.

PEKO benchmarks [8][9] are synthetic benchmarks with pre-known optimal solutions. All nets in PEKO benchmarks are local. We implemented a PEKO benchmark generator based on the algorithm described in [8][9] to produce PEKO benchmarks with different net degree distributions.

Syn benchmarks are also synthetic benchmarks with different Rent's exponents generated by the *gnl* [19] tool. Those benchmarks are built by recursive bottom-up clustering according to the Rent's rule. We obtained the gnl1.1.1 program from [24].

3. GLOBAL CHARACTERISTICS OF NETLISTS AND THEIR IMPACT ON PLACEMENT EFFICIENCY

Due to the heuristic nature of placement algorithms, they produce only sub-optimal solutions. In this section we study the global characteristics of netlist structures and try to correlate them with the placement efficiency of different placers.

3.1 Net degree distribution

We use a PEKO netlist generator to build three benchmark suites with different net degree distributions. Benchmarks in suite1 have only 2-pin nets. Benchmarks in suite2 have 2-pin and 3-pin nets in 4:1 ratio. Benchmarks in suite3 have 2-pin, 3-pin and 4-pin nets in 7:2:1 ratio. We place those benchmarks by Gold, Capo and mPL. The data are shown in Table 1 and are expressed as ratios r of the sum-of-half-bounding-boxes of nets in the placed and known optimal results:

$$r = \frac{W_S}{W_{opt}} \qquad \text{(EQ1)}$$

In each of the suites 1, 2 and 3 the benchmark test01 has 10,000 cells, test02 has 20,000 cells, and so on, up to 90,000 cells in test09. In these three suites the total net number in each circuit is 1.2 times the cell number. For example, test01 has 12,000 nets.

For each test case in Table 1, the three corresponding benchmarks in those suites have the same number of nets but different total pin numbers. From Table 1, we observe that increasing the count of multi-pin nets degrades the quality of placement produced by Capo and Gold on PEKO benchmarks. Or we can say that the net degree distribution has a significant impact on placement efficiency of Capo and Gold placers. Such impact is much smaller in the case of mPL.

We define r_n as a ratio of the average lengths of nets with degree n in the placed and optimal results:

$$r_n = \frac{AveL_n}{OptL_n} \qquad \text{(EQ2)}$$

In the above equation, $AveL_n$ is the average length of n-pin nets in the placement. The $OptL_n$ is the average length of n-pin nets in the optimal solution. To examine further the impact of net degree distribution on the placement efficiency of Capo and Gold, we show the optimal ratios of nets with degrees 2, 3 and 4 in benchmark suite3. These data are shown in Table 2. We observe that in the placement results generated by Capo and Gold, nets with larger degrees usually have a larger optimal ratio.

Table 2. Optimal ratio of nets with degree 2, 3, 4 in benchmark suite3

bench	Capo			Gold		
	2	3	4	2	3	4
test01	1.49	1.66	2.26	1.34	1.42	1.82
test02	1.56	1.79	2.46	1.40	1.51	2.00
test03	1.56	1.79	2.45	1.41	1.53	2.02
test04	1.55	1.75	2.39	1.40	1.51	1.96
test05	1.58	1.82	2.48	1.43	1.54	2.03
test06	1.61	1.85	2.57	1.45	1.58	2.12
test07	1.57	1.82	2.51	1.42	1.55	2.08
test08	1.60	1.85	2.58	1.44	1.58	2.12
test09	1.64	1.95	2.70	1.47	1.66	2.25
Ave	1.57	1.81	2.49	1.42	1.54	2.04

From the above discussion, we note that compared with 2-pin nets, multi-pin nets are more difficult to optimize by partition-based placer Capo and simulated-annealing-based placer Gold. Net degree distribution of a netlist is an important factor which significantly affects the placement efficiency of these two placers. Meanwhile the placement efficiency of the analytic placer mPL is relatively insensitive to the net degree distribution.

3.2 Net count

In this section, we experiment with a different PEKO benchmark suite which is composed of only 2-pin nets. Circuits in this suite each have 30,000 cells, but a different number of nets. In Table 3 we compare placement results obtained by Capo, Gold and mPL, and report them relative to the optimal placement. We plot these data in Figure 1.

Table 3. PEKO benchmarks with different net count

Bench	Capo	Gold	mPL
test_1(1.2)	1.406	1.304	1.359
test_2(1.3)	1.523	1.411	1.420
test_3(1.4)	1.608	1.476	1.462
test_4(1.5)	1.686	1.549	1.425
test_5(1.6)	1.804	1.636	1.466
test_6(1.7)	1.837	1.667	1.431
test_7(1.8)	1.886	1.696	1.440
test_8(1.9)	1.897	1.687	1.434
test_9(2.0)	1.855	1.619	1.457
test_10(2.1)	1.880	1.660	1.447

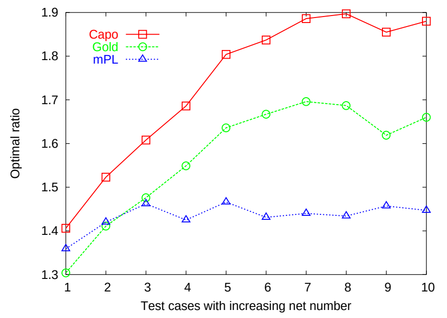

Figure 1: Net number VS Placement efficiency

In Table 3, the number in parenthesis next to the benchmark name denotes the ratio between the net and cell count in the circuit. Analyzing Table 3 and Figure 1, we note: (i) Gold and mPL obtain better placement results than Capo in all cases; (ii) With the increase of net count, placement efficiency of Capo and Gold decrease significantly at the beginning and saturate after the net count has passed a certain threshold; (iii) Placement efficiency of mPL is relatively insensitive to the net count.

3.3 Rent's exponent

The average number of terminals required to connect the gates in a module to the rest of the circuit and its exterior can be expressed by the following equation:

$$T = tG^p \qquad \text{(EQ3)}$$

Various interpretations of this power law, also known as Rent's rule, exist. The coefficient *t* and exponent *p* are referred to as the *Rent coefficient* and the *Rent exponent*. There is a good correlation between the Rent's exponent and interconnect complexity of circuits. Most of the real circuits have their Rent's exponents in the range of 0.2-0.8.

Table 4. Placement results of circuits with Rent's exponent from 0.2 to 0.8

p	0.2	0.3	0.4	0.5	0.6	0.7	0.8
Capo	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Dragon	1.242	1.161	1.089	1.030	0.975	0.946	0.931
mPL	0.948	0.962	0.972	0.983	1.014	0.969	0.992

We use gnl to build eight benchmarks with Rent's exponent 0.2 through 0.8. All these benchmarks have 40,000 cells and the same number of nets. Then we place these benchmarks by Capo, Dragon and mPL. The results, as shown in Table 4, are normalized to Capo. In the first row of the table, *p* denotes Rent's exponent of the circuit. The same data are plotted in Figure 2.

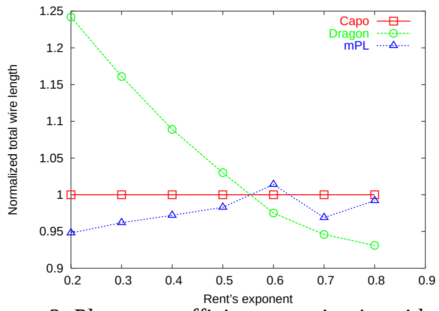

Figure 2: Placement efficiency on circuits with different p

From the experimental results we conclude that: (i) Dragon works very poorly on low Rent's exponent circuits. When Rent's exponent reaches 0.6 or is larger, Dragon outperforms the other two placers. In the final stage of global placement, Dragon uses simulated annealing to refine placement quality. The layout is partitioned into grids with preset grid size, and cells are snapped into grids. In Dragon, the average number of cells in one grid location is 3-7. During the annealing process (swapping or moving the cells between grids), the wire length estimation is made based on the assumption that all the cells are in the grid centers. As a result, the estimated wire length will have some errors. For low Rent's exponent circuits, the number of errors can become very large, because in such circuits many nets are short. The wire-length estimation error implies wrong decisions on cell-swapping or move-acceptance during simulated annealing. This in turn causes poor placement results after legalization (overlap elimination). To support this claim, in Table 5 we compare the estimated total wire length before legalization to the actual total wire length after legalization. All data are extracted from the Dragon's standard output.

Table 5. Comparing the estimated total wire lengths in global placement with actual total wire length after legalization

p	0.2	0.3	0.7	0.8
b_legal	4.05e+06	4.75e+06	1.85e+07	2.21e+07
b_legal	4.05e+06	4.75e+06	1.85e+07	2.21e+07
err	39%	35%	9.8%	7.9%

In Table 5, *b_legal* denotes the estimated total wire length before legalization and *a_legal* denotes the actual total wire length after legalization. *err* is the wire length estimation error calculated as *err* = (*a_legal*-*b_legal*)/*a_legal*. We show only the two lowest and two highest Rent's exponent benchmarks. These data demonstrate that wire length estimation is very inaccurate for low Rent's exponent circuits. The same argument can explain why Dragon works poorly on Grids and PEKO benchmarks. These two benchmark suites are extreme because all nets are local in optimal solutions. When Dragon places such circuits, it encounters many wire length estimation errors that imply wrong decisions in the simulated annealing stage, which in turn lead to poor placement quality. When we adjusted the grid sizes in our Gold placer, it produced good placement results on both benchmark suites.

(ii) mPL works well on low Rent's exponent circuits. Analytic placers put much effort on eliminating overlaps between cells. This legalization process degrades placement quality. Since higher Rent's exponent circuits have more global nets than lower Rent's exponent circuits, high Rent exponent circuits have a natural tendency to produce more overlaps to be resolved. For this reason mPL works well on low Rent's exponent circuits. Similarly, this can explain why mPL works well on Grids and PEKO benchmarks. Those circuits have only local nets in their optimal solutions. We will return to this problem in our later discussion.

In [18], the authors showed that increasing grid size in simulated annealing-based global placement could reduce the solution space of placement problem. However, our experimental results presented in the previous section suggest that larger grid sizes may introduce a larger wire-length estimation error, which in turn may compromise the placement quality.To choose the grid size for best placement quality, we need to consider these two contradictory factors simultaneously. In [18], the authors tried different grid size in a quick annealing schedule and chose the one with the best placement result. Here, we look into the problem of grid size choice in simulated annealing-based global placement.

We use gnl to generate nine benchmark suites, suite01 through suite09. Each suite consists of eight benchmarks with Rent's exponent ranging from 0.2 to 0.8 and having the same number of cells and nets. Benchmarks in suite01 have 10,000 cells and 10,000 nets. Benchmarks in suite02 have 20,000 cells and 20,000 nets, and so on. Then we place those examples using our purely simulated-annealing-based placer Gold. We adjust the grid size during global placement and list the results in Table 6.

Table 6. Rent's exponent vs. grid size choice

p	0.2	0.3	0.4	0.5	0.6	0.7	0.8
1	1.0	1.0	1.0	1.0	1.0	1.0	1.0
2	1.070	1.054	1.034	1.019	0.996	0.995	0.993
3	1.115	1.081	1.053	1.027	1.012	0.991	0.989
4	1.158	1.121	1.080	1.044	1.018	1.000	0.988
5	1.231	1.171	1.116	1.071	1.037	1.009	0.994
6	1.303	1.227	1.154	1.092	1.056	1.021	1.008
7	1.346	1.272	1.185	1.113	1.067	1.029	1.010
8	1.410	1.311	1.229	1.136	1.080	1.035	1.016

In the table, the first column corresponds to grid size measured in area of average cells. The *best grid size* is the grid size that gives the best placement result. Data for each grid size have been normalized to those of grid size 1. All data are the averages of all benchmarks in a particular suite. We note that the best grid size increases when Rent's exponent of the placed circuit increases. This so because for a fixed grid size and increasing Rent's exponent, the error of wire-length estimation during global placement decreases. For a sufficiently large Rent's exponent, the benefit of increasing the grid size to reduce the solution space begins to dominate the placement quality. We don't consider the cell size variations in the above experiments. We assume that all cells in the circuits have the same size. Under this assumption, each swap during simulated annealing is legal. For circuits with mixed cell sizes that would not be the case so we should consider the relationship between the grid size and optimization flexibility. This explains why in Table 6 it looks as if the annealing placer favors small grid sizes.

4. PEKO BENCHMARKS EXTRACTED FROM IBM BENCHMARKS

In this section, we look into a special class of low Rent's exponent circuits, the PEKO benchmarks. We will experiment with the PEKO benchmarks published in [8][9]. Those circuits have the same number of nets and net degree distribution as the circuits in the IBM benchmark suite. As before, all nets in the examples in PEKO suite are local. The authors of [8][9] applied different placers on PEKO benchmarks and computed the ratios of the placed and optimal results. Based on those results, they concluded that the current placers produce solutions far below the optimum possible. In this section, we look into the intrinsic characteristics of PEKO benchmarks and compare them to the real-circuit benchmarks. In the following sub-sections, we denote the PEKO benchmark suite extracted

Table 7. Rent's exponent of I-PEKO benchmarks and corresponding IBM benchmarks

Bench	09	10	11	12	13	14	15	16	17	18
I-PEKO	0.398	0.393	0.384	0.397	0.372	0.392	0.401	0.400	0.406	0.420
IBM	0.617	0.665	0.648	0.690	0.666	0.685	0.648	0.659	0.700	0.676

from the IBM benchmarks as I-PEKO to distinguish it from the PEKO benchmark suites we used in Section 3.

First we will show that although I-PEKO benchmarks have exactly the same net degree distribution as IBM benchmarks, that doesn't mean that they have similar interconnect complexities which can be characterized by the Rent's exponent. In Table 7 we compare the Rent's exponents of IBM benchmarks to the corresponding I-PEKO benchmarks.

It is evident that the I-PEKO benchmarks have much smaller Rent's exponents compared to their corresponding IBM benchmarks. Even though we don't know what would be the optimal solution for the IBM benchmarks, we can conclude that such solutions will have much larger sum-of-net length than the optimal solutions of their counterparts in the I-PEKO suite.

We define *good solutions* as solutions near the optimum solution. Each I-PEKO benchmark has only local nets. In the optimum solution, cells connected by a net are ideally packed together. It is apparent that such benchmarks with only local nets should have a smaller space for good solutions than other benchmarks have.

We build a PEKO circuit and a random circuit, each with 9 cells. So each circuit has 9! possible placements. Then we randomly generate ten thousand placement configurations for each circuit. The placement quality distributions are shown in Table 8.

Table 8. "good solution" distribution

Twl	opt	<=+ 1	<=+ 2	<=+ 3	<=+ 4	<=+ 5	<=+ 6	<=+ 7
random	1	13	85	236	669	1480	2670	4244
PEKO	1	6	26	82	191	435	944	1779

In the above table, the first row gives the placement quality in terms of total wire length. The second column *opt* gives the number of optimum solutions. <=*n* represents solutions with total wire length smaller or equal to optimal value plus n pitch size.

This small experiment shows that PEKO benchmarks have indeed a smaller *good solution* space than the random benchmarks have.

We use the Gold placer to refine the Capo and Dragon placement solutions on I-PEKO and IBM benchmarks. The results are shown in Table 9.

Table 9. Placement quality improvement

Benchmark	I-PEKO	IBM
Gold	1.0	1.0
Capo	1.261	1.151
Dragon	1.377	1.061

In Table 9 all the values are normalized with respect to Gold solutions. The numbers are the averages taken over all 18 benchmarks in the corresponding suite. The results indicate that the simulated-annealing-based refinement can improve the solutions of I-PEKO benchmarks generated by Capo and Dragon more than it can improve the solutions of IBM benchmarks.

Quadratic placer determines the placement solution by solving the following linear equation system

$$C \times p + f = 0. \qquad \text{(EQ4)}$$

In this equation, C is the connectivity matrix, p is the cell location vector, and f is the force vector generated by the fixed points (pads). Since such a solution in the equilibrium state does not consider overlaps, we need to eliminate them to get a legal solution. The traditional approaches to eliminate overlaps include density-induced force [10] and successive partitions [20].

Our quadratic placer consists of two steps. (i) Solve the linear equation system and get the equilibrium solution with overlaps. (ii) Apply simulated annealing to eliminate overlaps to get a legal placement. Since we use simulated annealing to find the legal placement, this quadratic placer only works well on benchmarks which need a small effort to eliminate overlaps. That is the reason why we do not experiment with it in Section 2.1. Discussion in the Section 3.3 shows that circuits with a large portion of local nets need minimal effort to eliminate overlap. The I-PEKO benchmarks have only local nets, so our quadratic placer can determine very good solutions for them. The results are shown in Table 10. The I-PEKO benchmark suite here has external pads, just as PEKO suite3 does in [8].

Table 10. Placement obtained by Quadratic and Gold

Bench	Gold		Quadratic	
	Opt	P_{layout}	Opt	P_{layout}
PEKO09	1.61	0.524	1.12	0.509
PEKO10	1.59	0.521	1.12	0.514
PEKO11	1.50	0.517	1.12	0.512
PEKO12	1.74	0.540	1.13	0.511
PEKO13	1.72	0.528	1.12	0.510
PEKO14	1.72	0.522	1.13	0.508
PEKO15	1.61	0.515	1.12	0.505
PEKO16	1.78	0.521	1.12	0.504
PEKO17	1.61	0.516	1.13	0.504
PEKO18	1.54	0.508	1.13	0.505
Ave	1.64	0.521	1.12	0.508

In Table 10, *Opt* denotes the interconnect length ratio in optimal and placed results, and p_{layout} denotes layout Rent's exponent of the corresponding placement. From these data we conclude that even compared to gold solutions, the quadratic placer gives very good placement results, which are already very near to the optimal solutions.

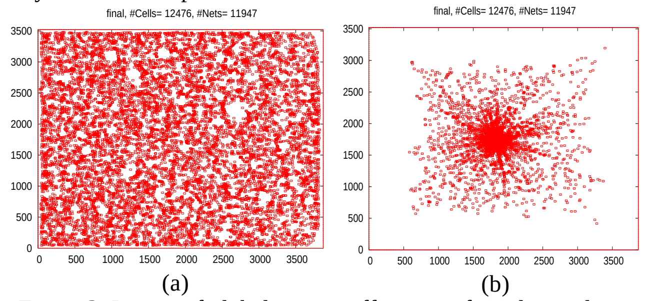

Figure 3: Impact of global nets on efficiency of quadratic placer

The equilibrium solution for PEKO01 is shown in Figure 3(a). It is clear that we need only a small additional effort to eliminate the overlaps. In [9], the authors introduce some randomly generated, non-local nets to the PEKO benchmark suite to mimic the effect of global nets. With a preset non-local ratio α, for i-pin nets, $\alpha \cdot d_i$ (d_i is the number of i-pin nets) nets are generated by randomly connecting i cells. The remaining nets are added as regular local nets. We build the G-PEKO01 circuit with the non-local ratio of 10% in the same way. The equilibrium solution of G-PEKO01 is shown in Figure 3(b). We observe that the more global nets we have the greater the effort needed to eliminate overlaps. This again confirms our discussion on analytic placers in Section 3.3.

5. CONCLUSIONS

In this paper, we have reported our study of the relationships between netlist structure and placement efficiency of different placement tools. For placement efficiency, we considered both placement quality and placement stability. We took three kinds of placement tools into account: analytic placer, simulated-annealing-based placer and partition-based placer. We used both real benchmarks and synthetic benchmarks to test these placers. Based on the observations and analyses of experimental results, we summarize several useful conclusions about relationships between the netlist structure and placement efficiency of placers: (i) Different placers favor netlists with different structural characteristics and show different sensitivity to the same characteristics. (ii) Placement efficiency correlates with interconnection complexity of circuits. (iii) Global nets increase the effort to eliminate overlaps of cells and degrade the placement quality of an analytic placer.

Future work will entail: (i) placement efficiency analysis of netlists with uneven interconnection complexity distribution; (ii) netlist restructuring targeting placement efficiency.

6. ACKNOWLEDGEMENTS

This work was supported by the California MICRO Program through grants from IBM and Cadence. The authors are indebted to Dr. Xiaojian Yang (Synplicity) and Mr. Bo-Kyung Choi (UCLA) for technical discussions.

7. REFERENCES

[1] S.N.Adya, M.C.Yildiz, I.L.Markov, P.G.Villarrubia, P.N.Parakh and P.H.Madden, "Benchmarking for Large-scale Placement and Beyond", in Proc. Intl. Symp. on Physical Design, pp.95-103, 2003.

[2] C.J.Alpert, J.H.Huang and A.B.Kahng, "Multi-level Circuit Partitioning", Design Automation Conference, pp.530-533, 1997.

[3] A.E.Caldwell, A.B.Kahng and I.L.Markov, "Improved Algorithms for Hypergraph Bisection", ASP-DAC, pp.661-666, 2000.

[4] A.E.Caldwell, A.B.Kahng and I.L.Markov, "Can Recursive Bisection alone Produce Routable Placements", Design Automation Conference, pp.260-263, 2000.

[5] A.E.Caldwell, A.B.Kahng and I.L.Markov, "Optimal Partitioners and End-case Placers for Standard-cell Layout", IEEE Transactions on Computer Aided Design,vol.19, no.11, pp.1304-1314, 2000.

[6] T. F. Chan, J. Cong, T. Kong and J. Shinnerl, "Multilevel Optimization for Large-scale Circuit Placement", Proc. of Int. Conf. on Computer-Aided Design, pp. 171-176, 2000.

[7] T. F. Chan, J. Cong, J. Shinnerl and K. Sze, "An Enhanced Multilevel Algorithm for Circuit Placement", Proc. of Int. Conf. on Computer-Aided Design, pp. 299-306, November 2003.

[8] C.C.Chang, J.Cong and M.Xie, "Optimality and Scalability Study of Existing Placement Algorithms", ASP-DAC, pp.621-627, 2003.

[9] J.Cong, M.Romesis and M.Xie, "Optimality and Scalability Study of Partitioning and Placement Algorithms", In Proc. Intl. Symp. on Physical Design, pp.88-94, Apr 2003

[10] H.Eisenmann and F.M.Johannes, "Generic Global Placement and Floorplanning", Design Automation Conference, pp.269-274, 1998.

[11] C.M.Fiduccia and R.M.Mattheyses, "A Linear Time Heuristic for Improve Network Partitions", Design Automation Conference, pp.175-181, 1982.

[12] B.Hu and M.Marek-Sadowska, "Fine-granularity Clustering for Large-scale Placement Problems", In Proc. Intl. Symp. on Physical Design, pp.67-74, Apr 2003

[13] G.Karypis, R.Aggarwal, V.Kurnar and S.Shekhar, "Multi-level Hypergraph Partition: Applications in VLSI Design", Design Automation Conference, pp.526-529, 1997.

[14] P.Kudva, A.Sullivan and W. Dougherty, "Metrics for Structural Logic Synthesis", Proc. of Int. Conf. on Computer-Aided Design, pp. 551-556, 2002.

[15] T.Kutzschebauch and L.Stok, "Congestion Aware Layout Driven Logic Synthesis", Proc. of Int. Conf. on Computer-Aided Design, pp. 216-223, 2001.

[16] M.Pedram and N.Bhat, "Layout Driven Technology Mapping", In Design Automation Conference, pp.99-105, 1991.

[17] M.Pedram and N.Bhat, "Layout Driven Logic Restructure/Decomposition", Proc. of Int. Conf. on Computer-Aided Design, pp. 134-137, 1991.

[18] M. Sarrafzadeh and M. Wang, "NRG: Global and Detailed Placement". Proc. of Int. Conf. on Computer-Aided Design, pp. 164-169, 1997

[19] D.Stroobandt, P.Verplaetse and J.Van Campenhout, "Generating Synthetic Benchmark Circuits for Evaluating CAD Tools", IEEE Transactions on Computer Aided Design,vol.19, no.9, 2000.

[20] R.Tsai, E.S.Kuh and C.Hsu, "PROUD: A Sea-of-gates Placement Algorithm", IEEE Design&Test of Computers, pp.44-56, 1988.

[21] M.Wang, X.Yang and M.Sarrafzadeh, "Dragon2000: Standard-cell Placement Tool for Large Industry Circuits", Proc. of Int. Conf. on Computer-Aided Design, pp. 260-264, 2000.

[22] X.Yang, B.K.Choi and M.Sarrafzadeh, "Routability Driven White Space Allocation for Fixed-Die Standard-Cell Placement", In Proc. Intl. Symp. on Physical Design, pp.42-47, Apr 2002.

[23] VLSI CAD Bookshelf: http://www.gigascale.org/bookshelf/

[24] Gnl Version 1.1.1: http://www.elis.ugent.be/~pvrplaet/gnl/