Optimal Placement by Branch-and-Price

Pradeep Ramachandaran[1] Ameya R. Agnihotri[2] Satoshi Ono[2,3,4]
Purushothaman Damodaran[1] Krishnaswami Srihari[1] Patrick H. Madden[2,4]
SUNY Binghamton SSIE[1] and CSD[2] FAIS[3] University of Kitakyushu[4]

***Abstract*— Circuit placement has a large impact on all aspects of performance; speed, power consumption, reliability, and cost are all affected by the physical locations of interconnected transistors. The placement problem is NP-Complete for even simple metrics.**

In this paper, we apply techniques developed by the Operations Research (OR) community to the placement problem. Using an Integer Programming (IP) formulation and by applying a "branch-and-price" approach, we are able to optimally solve placement problems that are an order of magnitude larger than those addressed by traditional methods. Our results show that suboptimality is rampant on the small scale, and that there is merit in increasing the size of optimization windows used in detail placement.

I. Introduction

System delay in modern circuits is dominated by the interconnect delay; by bringing interconnected devices closer together, we can improve performance.[1] For even simple metrics, the placement problem is intractable. Given n elements to place, there are $O(n!)$ different orderings; finding the optimal configuration is not practical for problems of even modest size. Placement is commonly divided in "global" and "detail" steps.

While much of the recent placement research has focused on better global placement methods, we find that detail placement offers a great deal of optimization potential. Our experiments have shown that for benchmarks with known optimal configurations, most global placement methods position cells close to their optimal locations[14]. Much of the wirelength suboptimality on these benchmarks, ranging from 30% to 150%, comes not from poor global placement, but from poor detail placement.

In this paper, we focus on new methods for detail placement. We are interested in finding optimal solutions for problems that are larger than those addressed by traditional methods. In traditional detailed placement, tools such as our *Feng Shui*[11] mixed size placement tool use branch-and-bound (B&B) to find optimal configurations for small portions of the circuit. Using a "sliding window" technique, an entire placement is improved by rearranging small groups of cells (usually less than six), one group at a time. Optimizing larger groups becomes prohibative, due to an explosion in run time.

Rather than using exhaustive enumeration or B&B to find optimal solutions for small groups, we instead apply a branch-and-price (B&P) approach to find *optimum* solutions for groups of up to 36 elements. The B&P approach is based on column generation techniques and B&B. B&P has been applied to solve large instances of well known NP-Complete problems such as the Vehicle Routing Problem [7].

We have tested our approach on benchmarks with known optimal configurations, and also on problems extracted from the "final" placements of a number of recent tools (*Feng Shui 2.0*, *Dragon 3.01*, and *mPL 3.0*). We find that suboptimality is rampant: for optimization windows of nine elements, roughly half of the test cases are suboptimal. As we scale towards windows with thirtysix elements, we find that roughly 85% of the test cases are suboptimal. The improvement possible from any single window is small–but collectively, we estimate that wirelengths can be reduced by as much as 30% with strong detail placement.

Our primary contributions are the identification of the seriousness of the detail placement problem, the study of the degree of suboptimality for different window sizes, and the development of an optimization approach that can handle window sizes far larger than traditional methods. The remainder of this paper is organized as follows. We first describe the traditional placement problem, and the specific formulation we have addressed with our tool. We next briefly survey prior work in placement, dividing this into "global" and "detail" placement. A consideration of optimality for placement completes our background discussion.

Section III describes the B&P approach; while this is well known in the OR community, we are not aware of its use within computer-aided design. We explain the approach for the problem under study. Section IV studies the use of our approach on benchmarks derived from industrial designs. We also compare the run times of B&P to traditional methods, showing that the technique allows a new class of problems to be effectively addressed.

We conclude the paper with a discussion of the results, and a brief description of our current work. We are actively working to integrate our approach with our placement tool, and are enhancing our formulation to improve run times, and to support logic elements with both differing widths and heights.

II. Previous Work

Placement is a well studied problem[16]. In placement, we have a large number of *logic elements*–traditionally, small blocks which implement basic boolean functions. We will use "cells" to indicate the individual logic elements, The cells are interconnected by a number of *signal nets*, or "nets." For simplicity, we focus on unit-sized cells.

[1]Interconnect delay depends on the topology, wire lengths, wire sizes, coupling capacitances, and driver strength, among other things. For simplicity, we focus only on half perimeter wire length, which is a common metric for placement.

Our objective is place a set of cells $c \in C$ into a set of locations $l \in L$, such that we minimize the total size of the bounding boxes for nets $n \in N$. Each net $n \in N$ connects a subset of cells. Overlap is not allowed, and $|L| \geq |C|$.

Global Placement

Simulated annealing[12], [17], [21] is a well known approach for circuit placement; while run times tend to be high, placement results are comparitively good. A second approach that has received increased attention recently is partitioning based placement, and particularly methods that use recursive bisection. Using modern multi-level partitioners, current bisection-based academic placement tools[4], [2] have obtained excellent results, with relatively low run times. A third popular approach is "analytic" placement[13], [9]. The placement problem is formulated as one of linear programming (LP); bounding box wire length is a convex function, and LP solvers are able to find orderings of cells to minimize bounding box, or bounding box squared objectives. Placements developed using analytic methods can be highly overlapping (a trivial optimal solution may be one in which all elements are placed on top of each other). To deal with the overlap, methods such as forced partitioning and repulsive forces have been implemented. Many current commercial placement tools utilize analytic methods.

A convergence on a set of basic metrics, and the availability of relatively large benchmarks, has shown that most current academic tools are within a few percent of each other in most cases. There is no clear "winner," with each approach having a slight advantage or disadvantage on a particular benchmark[1]. None of the placement tools are able to achieve optimality; in some cases, the degree of suboptimality is quite large.

Detail Placement

Following global placement, some techniques (particularly analytic and one form of recursive bisection) require "legalization." There may be initially some overlap in the cells, or cells may not be aligned with rows. There are a number of methods to legalize placements, ranging from flow-based[8], to dynamic programming[2], to a simple greedy heuristic[10].

Once a placement has been legalized, almost all current tools use "sliding window" optimization. Small groups of adjacent cells (usually less than six) are considered; all permutations of the group are evaluated, with the best permutation being selected. The number of permutations for a set of objects is exponential, and this limits the size of optimization windows.

The optimal placement method we describe here uses an IP formulation and a solution approach based on decomposition principles to solve larger optimization problems in reasonable amounts of run time.

Placement Suboptimality

It should be obvious that the difficulty of finding an optimal solution grows quickly with the problem size. If there are n cells and n locations, there are $O(n!)$ different possible assignments. For a 2×2 (3×3) optimization window with only 4 (9) elements the number of combinations to evaluate is 24 (362880). Although brute force methods could solve these smaller instances easily, windows with 16, 25, or 36 elements have 2.09e13, 1.55e25, or 3.71e41 combinations, respectively. Problems of this size cannot be solved by brute-force methods with acceptable run times.

The rapid growth in difficulty results in the use of heuristics for the "global" stage, with optimal methods only being used for very small "detail" problems. This division is illustrated in Figure 1. The division of work between heuristic and optimal methods is important. Heuristic methods will obviously generate suboptimal placements; minor suboptimalities can be eliminated by local improvement.

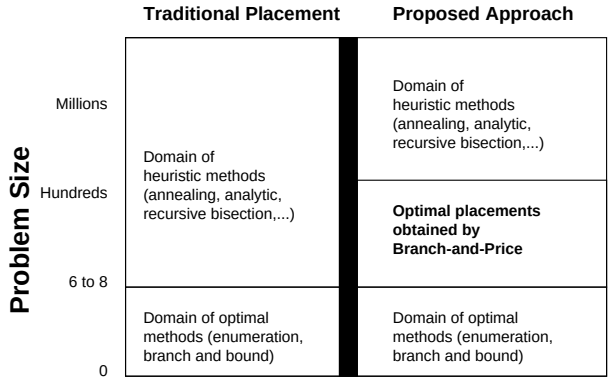

Fig. 1. Division between heuristic and optimal methods. The proposed approach allows for substantially larger problems to be solved optimally.

This work is motivated in part by recent studies on the suboptimality of placement algorithms. When traditional placement tools were applied on synthetic benchmarks with known optimal solutions, the results were from 50% to 150% away from optimal, indicating a large degree of suboptimality [5].

Given that placement results are known to be (significantly) suboptimal, it is reasonable to ask how much suboptimality is caused by global placement, and how much by detail placement. Figure 2 may be somewhat surprising: we present a histogram showing the number of cells a given Manhattan distance from a known optimal location. For the benchmark *PEKO01*, the placement tools *mPL*, *Feng Shui* (in both bisection and structural modes), and *Dragon* each place the vast majority of cells within twenty steps of their optimal locations[14]. **The global placements for this benchmark are extremely good; the majority of the suboptimality can be attributed to inadequate detail placement.**

III. Optimal Placement by B&P

Our approach to find optimal placements is effective because the IP formulation is embedded with special problem structure enabling us to decompose the large-scale placement problem into smaller subproblems. By identfying the special problem structure, a new tool to effectively solve the placement problem was devised. LP formulations for placement or not new. Our formulations to obtaining optimal placements is somewhat different from traditional placement tools like Gordian[13] or Kraftwerk[9]. Whereas Kraftwerk has variables to indicate the x and y coordinates of a particular cell,

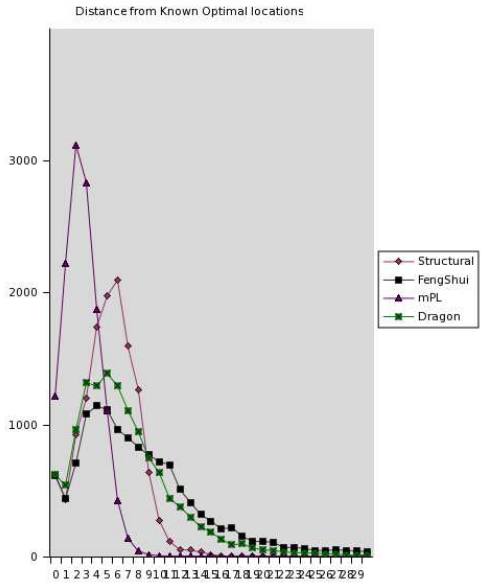

Fig. 2. Histogram of the number of cells a given distance from a known optimal location for the benchmark PEKO01. For the four placement tools listed, the vast majority of cells are within ten steps of their optimal location. Despite the good "global" placement, wire lengths are 30% to 50% higher than optimal.

our formulation contains variables corresponding to a *specific placement of all cells.* To be specific: any feasible assignment of cells to locations is a column; a binary variable takes a value of 1 if the assignment is chosen, and 0 otherwise.

If there are n cells, an IP or LP formulation would have $O(n!)$ columns; this is clearly impractical, both in terms of storage, and in the evaluation (pricing out) of columns. The Simplex procedure would require prohibitively long run times. Fortunately, decompostion techniques (such as Dantzig-Wolfe decomposition) can be applied to decompose the large problem into one or more smaller subproblems of manageable size.

The constraints in the subproblems usually posses a special problem structure, although this is not a requirement, that can be exploited. Binding the subproblems together is a "master problem" that ensures consistence across subproblems. By decomposing the problem, not all the columns are retained in the master problem. For this reason the master problem is generally called as a Restricted Master Problem (RMP). Improving columns are added to the RMP by solving the subproblems (a.k.a pricing problems). The entire problem is solved by iterating between the RMP and the subproblems; information is exchanged, and the process continues until no improving columns can be found. Thus, instead of retaining all the columns in the Simplex tableau, columns are added to the tableau through the search process. The procedure to add most promising columns is commonly referred to as the column generation approach in OR literature.

A. Dantzig-Wolfe Decomposition

The application of Dantzig-Wolfe decomposition to an LP problem is briefly described here; for an elaborate discussion we refer to [3]. Consider the following LP, where X is a polyhedral set representing constraints of special structure, A is an $m \times n$ matrix, and b is an m vector: minimize $\{cx|Ax = b, x \in X\}$. Let us assume that X is a bounded polyhedral set, then any point $x \in X$ can be expressed as a convex combination of the finite number of extreme points of X (i.e., $x = \sum_{j=1}^{t} \lambda_j x_j$, $\sum_{j=1}^{t} \lambda_j = 1$, and $\lambda_j \geq 0$, where x_j's are the extreme points of X). When X is unbounded, any point $x \in X$ can be expressed as a convex combination of the finite number of extreme points of X and a nonnegative combination of the extreme directions. The decomposition principle discussed next will apply for both the bounded and unbounded case. For simplicity, we discuss the bounded case.

Decomposing the problem and by substituting for x, the RMP is: minimize $\{\sum_{j=1}^{t}(cx_j)\lambda_j | \sum_{j=1}^{t}(Ax_j)\lambda_j = b, \sum_{j=1}^{t} \lambda_j = 1, \lambda_j \geq 0\}$. Since the number of extreme points of X is usually large, it is extremely difficult to explicitly enumerate these extreme points and solve this problem. At optimality, $cx_j - \alpha Ax_j - \beta = 0$ for all the basic variables and $cx_j - \alpha Ax_j - \beta \geq 0$ for all the non-basic variables, where α and β are the dual variables associated with the constraints in the RMP. Our objective is to determine the optimal solution without enumerating all the extreme points explicitly. In order to identify the most improving extreme point, we could solve the subproblem (SP): minimize $_{1 \leq j \leq t}$ $\{cx_j - \alpha Ax_j - \beta | x \in X\}$. The RMP is first solved and the dual values are used to define the objective function for the SP. The SP is then solved and an improving column is added to the RMP; the RMP is then reoptimized. An improving column is one which satisfies the condition $cx_j - \alpha Ax_j - \beta < 0$. When no improving column can be found, the solution is optimal. Thus, by not enumerating all the extreme points computational advantage over Simplex procedure is gained. For an overview of column generation approach, we ask the reader to refer to [20].

B. Branch-and-Price Technique

The B&P technique is based on the LP column generation approach and B&B. At root node of the B&B tree the column generation technique is applied to the linear relaxation of the RMP; this produces a lower bound. Solving the linear relaxation of the RMP does not guarantee an integer solution; cells may be partially assigned to multiple locations. The solution obtained, however, is a lower bound on *any* placement–this is used to guide a B&B approach. If no improving columns can be found and if the linear relaxation does not yield integer solution, branching is carried out on the fractional variables. Linear relaxations are easier to solve when compared to the IP formulations. The B&P technique applies column generation at each node in B&B tree to improve the lower bounds until an optimum is reached.

Decomposition techniques (especially B&P) have been successfully implemented to solve large-scale problems such as the Vehicle Routing [7], Airline Crew Scheduling [19], Generalized Assignment [15], and Scheduling [6], [18].Each of these studies identified the special structure embedded in the formulation, allowing decomposition of the original problem into a RMP and one or more SPs to successfully implement

column generation at each node of the B&B tree. In this research we formulate the cell placement problem and identify and exploit the special problem structure to devise efficient solution approach based on decomposition principle.

C. Formulation for Circuit Placement

We provide a detailed description of our mathematical formulation here. This formulation can be solved by commonly available MIP software packages; in our implementation, we use the commercial solver CPLEX.

Parameters:

$d_{ll'}$ distance between location l and l'
a_l distance between the left edge of chip and the center of location l
b_l distance between the top edge of chip and the center of location l

Decision Variables

$Y_{cln} = \begin{cases} 1, & \text{if cell } c \text{ in net } n \text{ is assigned to location } l \\ 0, & \text{otherwise.} \end{cases}$

$Z_{cl} = \begin{cases} 1, & \text{if cell } c \text{ is assigned to location } l \\ 0, & \text{otherwise.} \end{cases}$

p_n half perimeter of the rectangle enclosing all the cells in net n
A_n length of the rectangle enclosing all the cells in net n
B_n width of the rectangle enclosing all the cells in net n

Our model **PLACEMENT** is as follows.

$$Minimize \sum_{n\in N} p_n = \sum_{n\in N} (A_n + B_n) \quad (1)$$

Subject to

$$\sum_{c\in C} Z_{cl} = 1,\ \forall l \in L \quad (2)$$

$$\sum_{l\in L} Z_{cl} = 1,\ \forall c \in C \quad (3)$$

$$Y_{cln} \le Z_{cl},\ \forall l \in L,\ c \in C, n \in N \quad (4)$$

$$\sum_{c\in n} Y_{cln} \le 1,\ \forall n \in N,\ l \in L \quad (5)$$

$$\sum_{l\in L} Y_{cln} = 1,\ \forall n \in N,\ c \in n \quad (6)$$

$$\sum_{l'\in L} a_{l'} Y_{c'l'n} - \sum_{l\in L} a_l Y_{cln} \le A_n,\ \forall n \in N,\ c,\ c' \in n,\ c \ne c' \quad (7)$$

$$\sum_{l'\in L} b_{l'} Y_{c'l'n} - \sum_{l\in L} b_l Y_{cln} \le B_n,\ \forall n \in N,\ c,\ c' \in n,\ c \ne c' \quad (8)$$

$$Y_{cln}, Z_{cl} \in \{0,1\} \quad (9)$$

$$A_n,\ B_n,\ p_n \ge 0 \quad (10)$$

Objective (1) minimizes the total half perimeter wire length. Constraint (2) ensures that each location is occupied by exactly one cell and constraint (3) ensures that each cell is assigned to exactly one location. Constraint (4) ensures that cell c is assigned to location l for all the nets n in which cell c is a member. Constraints (5)-(9) correspond to each net. Constraint (5) ensures that each location is occupied by no more than one cell from net n. Constraint (6) ensures that each cell in a net is assigned to exactly one location. Constraints (7) and (8) determine the length and width of the rectangle that encloses all the cells in net n. Constraints (9) and (10) impose the binary and the non-negativity restrictions on the decision variables.

Model **PLACEMENT** can be represented in a general matrix form as shown below.

$$Minimize\ p_1 + \cdots + p_{|N|}$$

$$\begin{bmatrix} T_1 & & & \\ & T_2 & & \\ & & & T_{|N|} \\ A_1 & A_2 & & A_{|N|} \end{bmatrix} \approx \begin{bmatrix} R_1 \\ R_2 \\ \\ R_{|N|} \\ Q \end{bmatrix}$$

Where

T_n technological coefficient matrix of constraints (5) - (10) for net n
R_n right hand side coefficients for signal net n
A_n technological coefficient matrix for the linking constraints (2) - (4)
Q right hand side coefficients for linking constraints (2) - (4)
$\approx$ indicates either $\le$ or = depending on the constraint

1) Master Problem: From the matrix representation it is evident that model **PLACEMENT** has block diagonal structure (i.e., $T_i \cap T_j = \emptyset,\ \forall i, j \in N,\ i \ne j$). Hence, the optimum cell locations for individual nets can be solved independent of each other. This enables us to decompose the original problem into smaller sub-problems (one sub-problem for each signal net) by applying Dantzig- Wolfe decomposition. Letting λ_n^k denote the extreme points of the polytope associated with sub-problem n, and applying Dantzig-Wolfe decomposition to the model **PLACEMENT**, we obtain the RMP:

$$Minimize \sum_{n\in N} \sum_{k\in K_n} p_n^k \lambda_n^k \quad (11)$$

Subject to

$$\sum_{c\in C} Z_{cl} = 1,\ \forall l \in L \quad (12)$$

$$\sum_{l\in L} Z_{cl} = 1,\ \forall c \in C \quad (13)$$

$$\sum_{k\in K_n} Y_{cln}^k \lambda_n^k \le Z_{cl},\ \forall c \in C,\ l \in L,\ n \in N \qquad (\alpha_{cln}) \quad (14)$$

$$\sum_{k\in K_n} \lambda_n^k = 1,\ \forall n \in N \qquad (\gamma_n) \quad (15)$$

$$Z_{cl},\ \lambda_n^k \in \{0,1\} \quad (16)$$

2) Subproblems: Suppose α_{cln} and γ_n denote the dual variables associated with constraints (14) and (15), respectively. Decomposition results in an independent sub-problem for each signal net n. The sub-problem for signal net n:

$$Minimize\ \psi_n = p_n - \sum_{l\in L} \sum_{c\in n} \alpha_{cln} Y_{cln} - \gamma_n \quad (17)$$

Subject to constraints (5) - (10)

Linear relaxation (LR) of the RMP ($0 \le \lambda_n^k \le 1,\ 0 \le Z_{cl} \le 1$) is solved first and the dual variable values are used to define the objective (17) for the SP. The SP is solved and if an improving column is identified, the column is added to the RMP and reoptimized. An improving column is one that satisfies the criteria $\psi_n = p_n^* - \sum_{l\in L} \sum_{c\in n} \alpha_{cln} Y_{cln}^* - \gamma_n < 0$, where p_n^* and Y_{cln}^* are the optimal solution for signal net n. The column

added to the RMP provides the location information for each cell from different nets. The constraints in the RMP ensure that a cell in different nets is assigned to the same location. The optimal solution for the LR of RMP is at hand when no SP can generate an improving column. If the solution to the LR of the RMP yields some fractional λ or Z variables, branching occurs.

Branching relative to λ variables in the RMP would change the structure of the sub-problems and is not desirable [20]. So, branching should be preformed relative to the Z and Y variables. However, fixing Z variable to 0 or 1 would fix λ variable to 0 or 1. Consequently, branching was limited to Z variables. When a Z variable is fractional, the Z variable close to 0.5 (say Z_{cl})is chosen for branching. If Z_{cl} is fixed to zero (left child), all the columns with cell c in location l should be removed from the basis, the corresponding λ_{cln} should be set to 0, and new improving columns are added such that cell c is not assigned to location l. If Z_{cl} is fixed to one (right child), all the columns with cell c in locations other than l should be removed from the basis, corresponding λ_{cln} should be set to 0, and new improving columns are added.

IV. Experimental Results

We have performed experiments with mesh structures, which have known optimal configurations, and also with problems extracted from the final placements of a number of academic tools. For the mesh benchmarks, we start from a random initial position; our tool was able to find the optimal configurations in all cases, and with low run times. Results and run times are shown in Table I. Using a B&B solver, the 3×3 problems could be solved quickly, but the 4×4 problems took several minutes. Rough calculations indicate that the 5×5 problems would not complete with a year of run time. While B&B is faster than B&P for very small problems, larger problems are beyond the range of the traditional method.

To evaluate how common "local" suboptimality is in the results of current tools, we extracted subproblems from placements of the benchmark IBM01, using the tools Dragon 3.01[21], Feng Shui 2.0[2], and mPL 3.0[5]. All cell sizes were modified to be unit squares, which simplified the implementation of our prototype solver.

Nine sample problems were extracted from each placement and the problems ranged in size from 3×3 to 6×6. We wished to avoid "corner" cases where suboptimality might vary from the norm. In close to half of the 3×3 cases, and roughly 85% of the 6×6 cases, the placements of the academic tools were suboptimal. In some cases, run times are increased: high degree nets are not handled efficiently by our current implementation, and we plan to address this issue shortly. Results are shown in Table II.

All runs were performed under Windows on a 700mhz Pentium-based PC, and required a license for the commercial LP solver CPLEX. As reference placements were produced by Linux-based tools, running the experiments required manual intervention. Our current work involves adapting our tool to run under UNIX, and integration of the method into our placement tool, so that an entire design can be processed.

The wire lengths and improvements we report here are with respect to the potential improvement that can be obtained within the optimization window. For example, if a net spans outside of the optimization window, the portion of the bounding box that lies outside the window is not considered. For an $m \times m$ optimization window, we create a "ring" of dummy terminals to represent pins outside the window.

V. Summary and Conclusion

It is widely accepted that placement tools produce results that are far from optimal; further study has indicated that a great deal of this suboptimality is "local," and cannot be eliminated by improvements to global placement methods.

This has motivated our use of techniques from the operations research community. We can obtain optimal placements for large window sizes, and have shown that local suboptimality is rampant in the placements of current tools. By expanding the window further, and applying our technique across an entire design, we expect dramatic improvements.

There are a number of optimizations to the solver which we have yet to implement; we anticipate an order of magnitude or more improvement in run time through the use of an enhanced technique for handling subproblems with high degree nets. Our current work involves the integration of our method with our *Feng Shui* placement tool, and on addition of support for circuit elements of different sizes. Our current implementation uses the commercial LP solver CPLEX; we are considering the use of the IBM's open-source solver COIN, which would allow our tool to be distributed without a license restriction.

Acknowledgements: This work was supported in part by an IBM Faculty Partnership Award, SRC Project 947.1, an equipment grant from Intel, NYSTAR MDC, and by funds from the Japanese Ministry of ECSST via Kitakyushu and Fukuoka knowledge-based cluster project. We would like to thank our industry colleagues – particularly the research staff at IBM TJ Watson – for their comments and suggestions.

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. Int. Symp. on Physical Design*, pages 95–103, 2003.

[2] A. Agnihotri, M. C. Yildiz, A. Khatkhate, A. Mathur, S. Ono, and P. H. Madden. Fractional cut: Improved recursive bisection placement. In *Proc. Int. Conf. on Computer Aided Design*, pages 307–310, 2003.

[3] M. S. Bazaraa, J. Jarvis, and H. D. Sherali. *Linear Programming and Network Flows*. John Wiley & Sons, 2nd edition, 1990.

[4] A. E. Caldwell, A. B. Kahng, and I. L. Markov. Can recursive bisection alone produce routable placements? In *Proc. Design Automation Conf*, pages 477–482, 2000.

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

[6] Z. Chen and W. B. Powell. A column generation based decomposition algorithm for a parallel machine just-in-time scheduling problem. *European Journal of Operational Research*, 116(1):220–232, 1999.

[7] M. Desrochers, J. Desrosiers, and M. M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40(2):342–354, 1992.

3 by 3 Grid				4 by 4 Grid				5 by 5 Grid				6 by 6 Grid			
WL	Opt	%	RT (s)	WL	Opt	%	RT (s)	WL	Opt	%	RT (s)	WL	Opt	%	RT (s)
34	12	64.71	0.562	84	24	71.43	10.36	129	40	68.99	134.64	269	60	77.70	1407.66
30	12	60.00	0.5	72	24	66.67	12.89	151	40	73.51	170.33	263	60	77.19	1476.19
28	12	57.14	0.42	75	24	68.00	11.31	171	40	76.61	155.5	245	60	75.51	1721.33
34	12	64.71	0.55	81	24	70.37	12.12	156	40	74.36	164.08	241	60	75.10	1567.2
33	12	63.64	0.62	85	24	71.76	10.95	156	40	74.36	149.64	261	60	77.01	2007.3

TABLE I

EXPERIMENTS ON REGULAR MESH STRUCTURES; FOR GRID SIZES FROM 3 BY 3 TO 6 BY 6, OUR B&P APPROACH WAS ABLE TO FIND AN OPTIMAL SOLUTION, FROM ANY STARTING POINT. FOR EVEN A 4 BY 4 PROBLEM, THERE ARE 2.09×10^{13}, MAKING B&B OR ENUMERATION INFEASIBLE.

	3 by 3 Window				4 by 4 Window				5 by 5 Window				6 by 6 Window			
Benchmark	WL	Opt	%	RT (s)	WL	Opt	%	RT (s)	WL	Opt	%	RT (s)	WL	Opt	%	RT (s)
Dragon 1	33	33	0.000	0.125	103	103	0.000	0.39	185	182	1.622	269.703	259	255	1.544	39.406
Dragon 2	93	91	2.151	0.921	190	188	1.053	13.968	269	267	0.743	97.515	416	415	0.240	14.238
Dragon 3	117	117	0.000	0.375	186	185	0.538	1375.66	273	273	0.000	7200	386	386	0.000	7200
Dragon 4	88	88	0.000	0.187	169	169	0.000	3.046	273	273	0.000	4.187	430	426	0.930	528.375
Dragon 5	110	106	3.636	0.187	183	177	3.279	1.765	286	279	2.448	98.343	448	443	1.116	120.96
Dragon 6	107	107	0.000	0.156	190	189	0.526	3.766	290	288	0.690	243.765	378	378	0.000	7200
Dragon 7	133	133	0.000	0.265	194	194	0.000	8.562	280	279	0.357	18099.06	359	359	0.000	7200
Dragon 8	76	76	0.000	0.141	192	191	0.521	0.421	334	331	0.898	7.359	466	455	2.361	300.453
Dragon 9	145	145	0.000	0.375	211	211	0.000	6.937	271	270	0.369	6590.484	367	366	0.272	65.359
Feng Shui 1	88	84	4.545	0.328	203	195	3.941	7.39	394	382	3.046	570.48	569	569	0.000	7200
Feng Shui 2	81	73	9.877	0.312	203	187	7.882	2.14	286	270	5.594	72.234	404	397	1.733	7.3
Feng Shui 3	98	94	4.082	0.234	219	212	3.196	0.89	356	342	0.000	31.093	499	499	0.000	7200
Feng Shui 4	129	125	3.101	0.406	198	191	3.535	42.484	273	263	3.663	2966.765	351	351	0.000	7200
Feng Shui 5	59	58	1.695	4.031	115	111	3.478	80.25	182	177	2.747	727.64	250	247	1.200	6.687
Feng Shui 6	31	31	0.000	0.125	58	57	1.724	0.39	112	109	2.679	2.406	209	195	6.699	13.89
Feng Shui 7	120	116	3.333	11.88	205	200	2.439	58.765	370	365	1.351	309.812	562	552	1.779	3559.31
Feng Shui 8	103	102	0.971	1.156	202	196	2.970	4.218	304	288	5.263	1131.046	384	384	0.000	7200
Feng Shui 9	71	69	2.817	0.515	145	137	5.517	33.531	219	208	5.023	801.14	319	316	0.940	37.125
mPL 1	92	92	0.000	0.375	137	137	0.000	27.39	192	192	0.000	7200	256	256	0.000	7200
mPL 2	78	77	1.282	0.203	136	135	0.735	0.703	231	228	1.299	13.688	358	357	0.279	28.343
mPL 3	147	147	0.000	0.379	269	268	0.372	24.156	404	404	0.000	7200	543	543	0.000	7200
mPL 4	81	81	0.000	0.14	168	168	0.000	0.703	242	240	0.826	70.937	427	420	1.639	163.24
mPL 5	112	111	0.893	0.187	200	199	0.500	0.656	334	333	0.299	6.89	455	454	0.220	1793.73
mPL 6	63	63	0.000	0.265	134	133	0.746	0.812	250	248	0.800	4.844	438	433	1.142	1164.08
mPL 7	111	111	0.000	0.218	197	197	0.000	0.468	350	348	0.571	4.015	515	512	0.583	157.937
mPL 8	90	89	1.111	0.359	150	149	0.667	1.484	230	229	0.435	16.125	305	304	0.328	148.578
mPL 9	125	125	0.000	0.343	205	205	0.000	2.453	327	326	0.306	45.71	411	410	0.243	298.484

TABLE II

EXPERIMENTS ON PORTIONS OF PLACEMENTS PRODUCED BY THREE DIFFERENT ACADEMIC TOOLS. IN ALL CASES, INCREASING THE WINDOW SIZE RESULTED IN SUBOPTIMAL CONFIGURATIONS BEING FOUND. WIRE LENGTH (WL) CONSIDERS THE WIRING WITHIN THE OPTIMIZATION WINDOW, WHILE "OPT" INDICATES THE WIRE LENGTH OF AN OPTIMAL SOLUTION. THE GAP FROM OPTIMAL IS INDICATED IN "%".

[8] K. Doll, F. M. Johannes, and K. J. Antreich. Iterative placement improvement by network flow methods. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 13(10):1189–1200, 1994.

[9] H. Eisenmann and F. M. Johannes. Generic global placement and floorplanning. In *Proc. Design Automation Conf*, pages 269–274, 1998.

[10] D. Hill. US patent 6,370,673: Method and system for high speed detailed placement of cells within an integrated circuit design, 2002.

[11] A. Khatkhate, C. Li, A. R. Agnihotri, M. C. Yildiz, S. Ono, C.-K. Koh, and P. H. Madden. Recursive bisection based mixed block placement. In *Proc. Int. Symp. on Physical Design*, pages 84–89, 2004.

[12] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.

[13] J. Kleinhans, G. Sigl, F. Johannes, and K. Antreich. GORDIAN: VLSI placement by quadratic programming and slicing optimization. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 10(3):356–365, 1991.

[14] S. Ono and P. H. Madden. On structure and suboptimality in placement. In *Proc. Asia South Pacific Design Automation Conf.*, page to appear, 2005.

[15] M. W. P. Savelsbergh. A branch-and-price algorithm for the generalized assignment problem. *Operations Research*, 6:831–841, 1997.

[16] K. Shahookar and P. Mazumder. Vlsi cell placement techniques. *ACM Computing Surveys*, 23(2):143–220, June 1991.

[17] W. Swartz and C. Sechen. Timing driven placement for large standard cell circuits. In *Proc. Design Automation Conf*, pages 211–215, 1995.

[18] J. M. van den Akker, C. A. J. Hurkens, and M. W. P. Savelsbergh. Time indexed formulations for machine scheduling problems: Column generation. *INFORMS Journal of Computing*, 12(2):111–124, 2000.

[19] P. H. Vance, C. Barnhart, E. L. Johnson, and G. L. Nemhauser. Airline crew scheduling: A new formulation and decomposition algorithm. *Operations Research*, 45:188–200, 1997.

[20] W. E. Wilhelm. A technical review of column generation in integer programming. *Optimization and Engineering*, 2:159–200, 2001.

[21] X. Yang, B.-K. Choi, and M. Sarrafzadeh. Routability driven white space allocation for fixed-die standard cell placement. In *Proc. Int. Symp. on Physical Design*, pages 42–50, 2002.