Exact methods for the Strip Packing problem Research Report*

Péter Madarasi
ELTE, Pázmány Péter Sétány 1/c, Budapest

supervised by
Tamás Kis
MTA SZTAKI, Kende str. 13-17, Budapest

1 Introduction

In this work, the two-dimensional Strip Packing problem is considered, which consists in packing n rectangles on a strip with a given width, and an infinite height. The rectangles must be placed in a non-overlapping, orthogonal way respecting the width of the strip, and the objective is to minimize the height of the strip.

Let R be the set of the given rectangles, and let W be the width of the strip. The width and the height of the i-th item is denoted by a_i and b_i, respectively. Both W and the dimensions of the items are supposed to be integers.

The bottom left corner of the strip is designated to be the origin of a plane with axes x and y corresponding to the width and the height of the strip, respectively. A given packing can be represented by the location of the bottom left corner of each rectangle i, denoted by (x_i, y_i).

*This work was supported by the K112881 research grant of the National Research, Development and Innovation Office – NKFIH.

1.1 Notation

Table 1: Constants

n	number of rectangles
$i = 1..n$	the i-th rectangle
a_i	width of rectangle i
b_i	height of rectangle i
W	width of the strip
H_{lb}	best known lower bound on the height of the strip
H_{ub}	best known upper bound on the height of the strip
H_{inc}	incumbent height of the strip found so far

Table 2: Variables

h	variable for the height of the strip
x_i	horizontal position of the left bottom corner of rectangle i
y_i	vertical position of the left bottom corner of rectangle i
$zl_{ij}, i \neq j$	indicates if rectangle i is on the left of rectangle j
$zd_{ij}, i \neq j$	indicates if rectangle i is under rectangle j
$sl_{ij}, i \neq j$	indicates if rectangle i is on the semi left of rectangle j
$sr_{ij}, i \neq j$	indicates if rectangle i is on the semi right of rectangle j
$sd_{ij}, i \neq j$	indicates if rectangle i is semi under rectangle j
$su_{ij}, i \neq j$	indicates if rectangle i is semi above rectangle j
$hb_{ij}, i \neq j$	indicates if rectangle j is in the row of rectangle i
$vb_{ij}, i \neq j$	indicates if rectangle j is in the column of rectangle i

Table 3: Functions, abbreviations

$\tilde{g}(Z)$	$\sum_{z \in Z} g(z)$, where $g : Z \longrightarrow \mathbb{R}$
$i = 1..n$	the i-th rectangle

2 MIP model

In this section a MIP model is described for the proposed problem.

$$
\begin{array}{lll}
\text{minimize} & h & \\
\text{subject to} & x_i + a_i \leq W, & i = 1..n \\
& y_i + b_i \leq h, & i = 1..n \\
& x_i, y_i \geq 0, & i = 1..n \\
& x_i + a_i \leq x_j \ OR \ x_j + a_j \leq x_i \ OR & \\
& \quad y_i + b_i \leq y_j \ OR \ y_j + b_j \leq y_i, & i = 1..n, j = 1..n
\end{array}
$$

The last set of constraints force the rectangles to be disjoint, and can be incorporated to the model by using binary variables as follows.

$$
\begin{array}{lll}
\text{minimize} & h & \\
\text{subject to} & x_i + a_i \leq W, & i = 1..n \\
& y_i + b_i \leq h, & i = 1..n \\
& x_i + a_i \leq x_j + W(1 - zl_{ij}) & i = 1..n,\ j = 1..n,\ (i \neq j) \\
& y_i + b_i \leq y_j + H_{ub}(1 - zd_{ij}) & i = 1..n,\ j = 1..n\ (i \neq j) \\
& zl_{ij} + zl_{ji} + zd_{ji} + zd_{ij} \geq 1 & i = 1..n,\ j = i + 1..n\ (i \neq j) \\
& x_i, y_i \geq 0, & i = 1..n \\
& zd_{ij}, zl_{ij} \in \{0, 1\}, & i = 1..n,\ j = i + 1..n\ (i \neq j)
\end{array}
$$

where H_{ub} is an upper bound of the height of the strip (e.g. $H_{ub} = \sum_{i \in R} b_i$ is clearly a proper choice).

3 Heuristics

To gain a valid tighter upper bound of the height of the strip than the one mentioned above, a few packings are generated using heuristic algorithms implemented by Balu. Note that no warm start solution is provided to CPLEX, but only the height of the best solution found preliminary. Not only does a tight upper bound provide an upper bound on variable h, but is allows the "BigM's" to be set smaller (see section 4.2.3).

4 Speeding up CPLEX

This section summarizes the current and some of the attempted improvements implemented to speed up the B&B procedure of CPLEX.

4.1 Initial lower bounds

4.1.1 Area lower bound

Obviously, $H_{lb} := \lceil \sum_{i \in R} a_i b_i / W \rceil$ is an easy to calculate lower bound.

4.1.2 Covering lower bound

Let K be the (arbitrarily ordered) set of the rectangle subsets having cumulative width at most W. Let χ_i ($i = 1..|K|$) denote the incidence vector of the i-th set of K, and let $b := (b_1, b_2, .., b_n)$ be the vector of heights of the rectangles. The nearest rounded up value of the optimum of the following linear program provides a lower bound of the optimal strip height.

$$\begin{array}{ll} \text{minimize} & \mathbf{1}\alpha \\ \text{subject to} & \sum_{i=1}^{|K|} \alpha_i \chi_i \geq b, \ \alpha \geq 0 \end{array}$$

This problem can be considered as a column generation problem, in which the subproblem to be solved is the knapsack problem. In the recent implementation, a dual cutting plane method is used, and the separation problems are solved with CPLEX. Based on preliminary tests, this approach provides a much tighter lower bound than the Area lower bound does (see section 4.1.1). Furthermore, it consistently excels the lower bound CPLEX can reach in a few seconds. In some lucky cases, the found lower bound is the optimum itself.

4.2 Tightening the relaxation

In each node of the B&B tree, an LP-relaxation of the current (restricted) problem is solved, and thus a lower bound of the best solution of the current branching is found, which might ensure that no better solution can be found on the current branch than the incumbent one. In this case, the current branch can be pruned without further investigation. Clearly, the tighter the LP relaxation is, the better lower bound is found, which is crucial to recognize the unfruitful branches immediately. Thus in the following, an extended model and some methods (i.e. valid cuts) strengthening the LP relaxation are described.

4.2.1 Extended MIP model

The model described in section 2 is extended with new variables to have the opportunity to introduce further cuts. Rectangle i is said to be on the *semi left* of rectangle j iff $x_i + a_i \leq x_j + b_j$. The other semi directions are defined similarly. Let $sl_{ij}, sr_{ij}, sd_{ij}, su_{ij}$ indicate whether rectangle i is on the semi left, right, down, up of rectangle j, respectively. To express these connections, the following inequalities are added.

$$\begin{array}{ll}
x_i + a_i \leq x_j + a_j + W(1 - sl_{ij}), & i = 1..n,\ j = 1..n,\ (i \neq j) \\
x_j \leq x_i + W(1 - sr_{ij}), & i = 1..n,\ j = 1..n,\ (i \neq j) \\
y_i + b_i \leq y_j + b_j + H_{ub}(1 - sd_{ij}), & i = 1..n,\ j = 1..n,\ (i \neq j) \\
y_j \leq y_i + H_{ub}(1 - su_{ij}), & i = 1..n,\ j = 1..n,\ (i \neq j) \\
sl_{ij}, sr_{ij}, sd_{ij}, su_{ij} \in \{0, 1\} & i = 1..n,\ j = 1..n,\ (i \neq j)
\end{array}$$

Furthermore, hb_{ij} and vb_{ij} indicate whether rectangle j is in the row of rectangle i, and whether rectangle j is in the column of rectangle i, respectively. Clearly $hb_{ij} = su_{ji}sd_{ji}$ and $vb_{ij} = sl_{ji}sr_{ji}$, which are easy to linearize.

4.2.2 Compelling the decision variables to be positive

So far binary variables have been introduced to force certain equalities if set, e.g. setting zl_{ij} to 1 ensures that rectangle i is on the left side of j. To strengthen the relaxation, the reverse should be prescribed as well, that is all these variables should be one if possible. For instance, if i is located on the left of j, then zl_{ij} should be one, i.e. $x_i + a_i \leq x_j + a_j \Longrightarrow zl_{ij} = 1$, which just the same as $zl_{ij} = 0 \Longrightarrow x_i + a_i > x_j + a_i$. With all the dimensions of the rectangles being integers, the latter is the same as $Wzl_{ij} + x_i + a_i \geq x_j + a_i + 1$. Applying the same idea to each and every binary variable, the following equalities are gained.

$$\begin{array}{ll}
x_j + 1 \leq x_i + a_i + Wzl_{ij}, & i = 1..n,\ j = 1..n,\ (i \neq j) \\
y_j + 1 \leq y_i + b_i + H_{ub} * zd_{ij}, & i = 1..n,\ j = 1..n,\ (i \neq j) \\
y_j + b_j + 1 \leq y_i + b_i + H_{ub}sd_{ij}, & i = 1..n,\ j = 1..n,\ (i \neq j) \\
y_i + 1 \leq y_j + H_{ub}su_{ij}, & i = 1..n,\ j = 1..n,\ (i \neq j) \\
x_j + a_j + 1 \leq x_i + a_i + Wsl_{ij}, & i = 1..n,\ j = 1..n,\ (i \neq j) \\
x_i + 1 \leq x_j + Wsr_{ij}, & i = 1..n,\ j = 1..n,\ (i \neq j)
\end{array}$$

4.2.3 Adjustment of BigM's

To tighten the relaxed polytope, BigM's should be chosen as small as possible. For instance, $x_j + 1 \leq x_i + a_i + Wzl_{ij}, i = 1..n,\ j = 1..n,\ (i \neq j)$ could be replaced by the following inequality.

$$x_j + 1 \leq x_i + a_i + (W - a_j - a_i + 1)zl_{ij}, \quad i = 1..n,\ j = 1..n,\ (i \neq j)$$

The same idea can be applied to all inequalities by using flag *-adjM*.

4.2.4 Row cuts

In the relaxed solution, there might exist a set S consisting of rectangles sharing the same row and having too large cumulative width. With S not respecting the width of the strip, no packing exists in which the items of S are located in the same row, i.e. at least one of the zd_{ij} variables among the rectangles of S must be one. Similarly, at most $\binom{|S|}{2} - 1$ of the zl_{ij} variables among rectangles of S can be one. That is, the following cuts are valid.

$$\sum_{\substack{i,j \in S \\ i \neq j}} zd_{ij} \geq 1$$

$$\sum_{\substack{i,j \in S \\ i \neq j}} zl_{ij} \leq \binom{|S|}{2} - 1$$

4.2.5 Column cuts

If the relaxed solution contains an item set S having larger cumulative height than the best upper bound of the necessary height of the strip, and the rectangles in S share the same column, no packing can include S in a column, i.e. the following holds for each and every packing.

$$\sum_{\substack{i,j \in S \\ i \neq j}} zl_{ij} \geq 1$$

$$\sum_{\substack{i,j \in S \\ i \neq j}} zd_{ij} \leq \binom{|S|}{2} - 1$$

4.2.6 Area cuts

Let $i \in R$ be a fixed item. Clearly, the sum of areas of rectangles located under rectangle i can not be larger than the area under i. That is,

$$\forall i \in R : \sum_{\substack{j \in R \\ i \neq j}} a_j b_j zd_{ji} \leq W y_i$$

is a valid cut. The same holds for the area above item i, and what is more, analogue equalities can be parsed on the rectangles located on the left and on the right of i.

$$\forall i \in R : \sum_{\substack{j \in R \\ i \neq j}} a_j b_j z d_{ij} \leq W(h - y_i - b_i)$$

$$\forall i \in R : \sum_{\substack{j \in R \\ i \neq j}} a_j b_j z l_{ji} \leq H_{inc} x_i$$

$$\forall i \in R : \sum_{\substack{j \in R \\ i \neq j}} a_j b_j z l_{ij} \leq H_{inc}(W - x_i - a_i)$$

Where H_{inc} denotes the height of the incumbent solution, or the best upper bound on the height of the optimal packing.

4.2.7 How to find row cuts

Suppose that S is a set of rectangles sharing the same row, and $\tilde{a}(S) > W$. The following equality described in Section 4.2.4 holds for every admissible rectangle packing.

$$\sum_{\substack{i,j \in S \\ i \neq j}} z d_{ij} \geq 1$$

Observe, that for any $S' \subseteq S : \tilde{a}(S') > W \implies \sum_{\substack{i,j \in S' \\ i \neq j}} z d_{ij} \geq 1$ holds for any admissible packing.

This section addresses the selection of such an S'.

To gain the strongest row cut, a set $S' \subseteq S$ should be determined, for which $\tilde{a}(S') > W$ and $\sum_{\substack{i,j \in S' \\ i \neq j}} z d_{ij}$ is as small as possible. This problem can be modelled as an integer program. To solve this integer program at each and every row cut generation, use the *-cut exCuts* flag.

This approach generates indeed strong cuts, but it is time-consuming. To balance the strength of the cut and the necessary time to find it, a heuristic method is presented. Let S' denote the already selected rectangles.

Definition 4.2.1. *The efficiency of a rectangle $i \in S \backslash S'$ is*

$$e_{S'}(i) = \begin{cases} a_i / \sum\limits_{j \in S'} (z d_{ij} + z d_{ji}) & \textit{if } \sum\limits_{j \in S'} (z d_{ij} + z d_{ji}) \neq 0, \\ \infty & \textit{otherwise.} \end{cases}$$

Algorithm 1 Efficient Subset

procedure EFFICIENTSUBSET
  $\Gamma_{zd}(j) := \sum_{i=1..n,\ i \neq j} zd_{ij} + zd_{ji}$
  $j^* \in \arg\max\{\Gamma_{zd}(j) : j = 1..n\}$
  $S' := \{j^*\}$
  while $\tilde{a}(S') \leq W$ **do**
    $j^* \in \arg\max\{e_{S'}(j) : j = 1..n\}$
    $S' := S' \cup \{j^*\}$
  Output S'

Algorithm 2 Local search

procedure LOCALSEARCH
  $\Gamma_{zd}^{S'}(j) := \sum_{i \in S' \setminus \{j\}} zd_{ij} + zd_{ji}$
  $\delta_{zd}^{S'}(i,j) := \Gamma_{zd}^{S'}(j) - \Gamma_{zd}^{S'}(i) - zd_{ij} - zd_{ji}$
  while $\min\{\delta_{zd}^{S'}(i,j) : i \in S', j \notin S', \tilde{a}(S') - a(i) + a(j) > W\} < 0$ **do**
    $(i^*, j^*) \in \arg\min\{\delta_{zd}^{S'}(i,j) : i \in S', j \notin S', \tilde{a}(S') - a(i) + a(j) > W\} < 0$
    $S' := S' \cup \{j^*\} \setminus \{i^*\}$
  Output S'

The designed method generates a set S' using Algorithm 1, and runs Algorithm 2 on this S'. To apply this cut generation procedure, use flag *-cut mostEff*.

4.2.8 Semi area cuts

Let $i \in R$ be a fixed item. Clearly, the sum of areas of rectangles located semi under rectangle i can not be larger than the area under the top of i. That is,

$$\forall i \in R : \sum_{\substack{j \in R \\ i \neq j}} a_j b_j sd_{ji} \leq W(y_i + b_i) - a_i b_i$$

is a valid cut. Similar considerations hold for the area semi above, left, right of item i.

$$\forall i \in R : \sum_{\substack{j \in R \\ i \neq j}} a_j b_j su_{ji} \leq W(h - y_i) - a_i b_i$$

$$\forall i \in R : \sum_{\substack{j \in R \\ i \neq j}} a_j b_j sl_{ji} \leq H_{inc}(x_i + a_i) - a_i b_i$$

$$\forall i \in R : \sum_{\substack{j \in R \\ i \neq j}} a_j b_j sr_{ji} \leq H_{inc}(W - x_i) - a_i b_i$$

Where H_{inc} denotes the height of the incumbent solution, or the best upper bound on the height of the optimal packing.

4.2.9 Belt cuts

Clearly, the sum of areas of rectangles located completely in the row of rectangle i can not be larger than $Wb_i - a_i b_i$, i.e. the area of the row of rectangle i minus the area of rectangle i. That is, the following inequation, called horizontal belt cut, holds for any admissible packing.

$$\forall i \in R : \sum_{\substack{j \in R \\ i \neq j}} a_j b_j \ vb_{ij} \leq Wb_i - a_i b_i$$

Similar observation leads to the following cut, called vertical belt cut.

$$\forall i \in R : \sum_{\substack{j \in R \\ i \neq j}} a_j b_j \ hb_{ij} \leq ha_i - a_i b_i$$

4.3 Branching rules

If the current branching could not be proved to include no better solution than the incumbent one, then new branches are constructed.

4.3.1 Braching priority

Intuition suggests that the efficiency of a packing is crucially influenced by the relative position of the largest rectangles, thus it seems to be reasonable to branch first on variables belonging to rectangles with large areas. CPLEX provides an interface to assign branching priorities to each non continuous variable, thus the above idea can be easily added to the B&B method by properly setting the priorities. Developing further this idea leads to a way more efficient custom branching rule detailed in the next section.

4.3.2 Custom branching based on the relaxed solution

To invalidate the current relaxed solution as much as possible, the branching is done on a variable between two rectangles having the largest intersecting area. After choosing the two mentioned rectangles i, j, two branchings are made. In the first one, a relative position of i, j is fixed, while in the other one the same position is forbidden. The positions are enumerated in the following order: $zl_{ij}, zl_{ji}, zd_{ij}, zd_{ji}$. That is, we try to avoid forcing one of the rectangles above another one. To apply this method, use flag *-bt maxInters*.

4.3.3 Custom weighted branching based on the relaxed solution

The Custom branching based on the relaxed solution described is section 4.3.2 is modified as follows. So far the order of zl_{ij}, zl_{ji} was determined without any consideration, this time let's prefer the one forcing less shifting. Furthermore, the branching where a variable is fixed gives a more restricted subproblem, so it should be preferred to the branching forbidding a relative position. To gain this, the estimated objective value (which is the optimum value of the current relaxed problem) of the fixing branch is increased by a small ϵ. Just the same method is applied to the zd_{ij}, zd_{ji} variables. To apply this method, use flag *-bt maxIntersW*.

5 Further ideas, observations, and experiments

5.1 On lifting

In this section, the tightening of the Row cuts of section 4.2.4 is attempted. Solely the first Row cut type has been considered so far, i.e. that

$$\sum_{\substack{i,j \in S \\ i \neq j}} zd_{ij} \geq 1,$$

where $S \subseteq R$ is too wide rectangle set.

Choose a too wide $S \subseteq R$ rectangle set such that $|S| \geq 3$ and $S \backslash \{k_3\}$ is still too wide, where k_3 denotes the third widest rectangle in S. Let k_1 and k_2 denote the widest, and the second widest rectangle of S, respectively. We claim that the following equality holds for any integer solution.

$$\sum_{\substack{i,j \in S \\ i \neq j}} zd_{ij} - zl_{k_1 k_2} - zl_{k_2 k_1} \geq 1$$

Proof: Case 1: $zl_{k_1k_2} = 1$ or $zl_{k_2k_1} = 1$, i.e. k_1 is forced to be on the left of k_2 or k_2 is forced to be on the left of k_1. It's sufficient to see that in this case at least two of the zd variables can be set to one in every integer solution. Assume - on the contrary - that at most one zd variable can be set to zero. This means that the width of S shrinks at most with the width of k_3. But $S\backslash\{k_3\}$ was supposed to be too wide. Contradiction. [Since in any two wide set, in an integer solution there is a rectangle pair with $zd = 1$.]
Case 2: $zl_{k_1k_2} = 0$ and $zl_{k_2k_1} = 0$. The equation is the basic row cut, see Section 4.2.4.

5.2 On semi relaxation

Based on preliminary tests, the problem arising by relaxing the zl variables is easy to solve. What if in the nodes of the B&B tree not the LP-relaxed is solved, but we relax the zl variables only?

5.3 On non-LP-based lower bounds

A sketchy idea is to relax the problem to a parallel scheduling problem by splitting all the rectangles vertically into smaller rectangles having width one. Let these small rectangles be associated with jobs, and let the processing time the height of the corresponding original rectangle. After that, the optimum value of scheduling all the jobs on W identical parallel machines provides a lower bound to the Strip Packing problem. Furthermore, in each branching some precedence constraints are given, which can be incorporated to the scheduling problem too.

6 Practical evaluation on random instances

This section present the practical efficiency of the designed methods on two type of random instances. In all cases, the width of the strip was 100 units and the runtime limit was set to 600 seconds.

6.1 Thinner problemset

15 problem instances were generated, each consists of 13 rectangles having unique random integer widths and heights between 1 and 50. Table 4 shows the runtime results and the numbers of generated nodes, while Table 5 shows the lower bounds found using different methods.

6.2 Wider problemset

15 problem instances were generated, each consists of 14 rectangles having unique random integer widths and heights between 1 and 25. Table 6 shows the runtime results and the numbers of generated nodes, while Table 7 shows the lower bounds found using different methods. Note that no Scheduling lower bounds have been calculated, since the scheduling problems were harder to solve than the original one.

Table 4: Results on thinner rectangles

CPLEX def. [-imprsOff]		**-bt maxIntersW -cut cplexDef**		**-bt maxIntersW -cut mostEff**		**-bt maxIntersW -cut exCuts**		**-bt maxIntersW -cut mostEff -semiVarsOn**		**-bt maxIntersW -cut mostEff -semiVarsOn -adjM**		**-bt maxIntersW -cut cplexDef -semiVarsOn -adjM**	
Nodes	**Time (s)**	**Nodes**	**Time (s)**	**Nodes**	**Time (s)**	**Nodes**	**Time (s)**	**Nodes**	**Time (s)**	**Nodes**	**Time (s)**	**Nodes**	**Time (s)**
1938233	598.34	960710	598.31	98	0.39	177	33.95	131000	597.95	64	0.86	413990	598.3
51533	21.45	3905	2.69	254	0.6	105	11.72	209	2.08	583	3.99	1689	5.19
355622	108.96	23330	11.51	3590	6.34	83	19.05	13970	58.64	403	3.34	20439	32.83
202171	78.39	40602	23.94	10166	17.72	6267	2583.86	4107	21.9	42203	200.72	360409	598.68
0	0.02	0	0.05	0	0.04	0	0.04	0	0.13	0	0.2	0	0.19
0	0.02	0	0.12	0	0.07	10	3.14	0	0.21	50	0.86	20	0.59
92021	45.01	1049	0.92	521	1.59	167	31.46	514	4.69	1652	9.54	1790	5.24
0	0.04	30	0.12	0	0.06	50	16.09	30	0.68	140	1.9	0	0.41
272580	133.66	28869	26.15	58821	113.22	7408	2233	38995	207.57	22366	134	23295	59.59
39849	19.18	6725	5.8	3956	11.12	2873	850.19	7397	57.55	6536	46.4	5254	15.52
1610	0.47	120	0.2	85	0.34	40	17.29	140	1.75	20	0.68	270	1.85
0	0.02	0	0.08	47	0.25	49	19.01	130	1.45	140	1.78	0	0.46
0	0.04	20	0.12	50	0.39	40	17.48	50	0.81	70	1	30	0.8
0	0.04	190	0.29	110	0.47	91	32.4	440	4.32	209	2.23	720	3.34
137074	83.83	4147	2.68	2207	5.83	185	44.13	1097	7.75	740	5.78	2068	6.01
3090693	1089.47	1069697	672.98	79905	158.43	17545	5912.81	198079	967.48	75176	413.28	829974	1329

Table 5: Thinner LB

Opt. val.	Scheduling LB		Covering LB		Area LB	
	LB	Time (s)	LB	Time (s)	LB	Time (s)
31	30	21.6305	24	0.0791763	22	1.89E-07
35	35	124.832	28	0.455894	28	1.07E-07
26	26	20.4104	23	0.0887163	20	1.50E-07
28	26	23.9832	26	0.112041	24	1.90E-07
23	23	0.262276	23	0.0690082	14	1.36E-07
25	25	0.482481	25	0.0422273	15	1.23E-07
31	31	30.8373	26	0.421025	26	9.80E-08
25	25	0.4399	25	0.034967	19	1.23E-07
28	28	89.3987	26	0.408643	26	9.00E-08
23	22	93.2168	23	0.185502	22	1.63E-07
23	23	0.451309	23	0.0768258	20	1.51E-07
23	23	0.384421	23	0.0662252	19	1.22E-07
25	25	0.505587	25	0.062263	18	1.45E-07
21	21	0.477406	21	0.0618319	18	1.23E-07
34	34	28.8298	29	0.37802	29	1.76E-07

Table 6: Results on wider rectangles

CPLEX def. [-imprsOff]		-bt maxIntersW -cut cplexDef		-bt maxIntersW -cut mostEff		-bt maxIntersW -cut exCuts		-bt maxIntersW -cut mostEff -semiVarsOn		-bt maxIntersW -cut mostEff -semiVarsOn -adjM		-bt maxIntersW -cut cplexDef -semiVarsOn -adjM	
Nodes	**Time (s)**	**Nodes**	**Time (s)**	**Nodes**	**Time (s)**	**Nodes**	**Time (s)**	**Nodes**	**Time (s)**	**Nodes**	**Time (s)**	**Nodes**	**Time (s)**
111183	169.12	15412	35.23	43738	103.82	12192	1605.88	39937	301.52	44356	308.62	42181	274.76
1874	0.64	578	0.54	180	0.64	195	47.18	180	2.02	202	2.41	164	1.31
88584	34.88	7870	5.58	6071	9.27	5533	601.7	11862	51.4	11640	48.95	12986	26.25
138755	171.25	28560	61.24	29604	77.18	11510	1965.85	43793	329.01	58080	544.27	47803	262.32
14302	9.56	3629	5.35	2704	7.29	2258	347.15	5532	46.74	5977	46.21	5178	29.4
46845	40.54	10380	16.68	7743	18.37	5530	926.78	8166	57.7	9186	67.94	12173	49.6
8519	6.29	5858	7.66	12078	26.1	8756	669.77	10864	69.84	9480	64.98	9259	37.07
4474	3.68	826	1.54	499	1.7	497	72.64	784	7.84	1134	10.91	1504	8.16
386058	598.66	312338	598.73	267800	598.19	14076	1817.48	76507	582.36	78800	593.46	137286	597.91
577070	598.67	319559	598.7	230687	598.12	11821	2102.21	69144	590.41	74498	597	133223	597.97
392011	472.95	29137	45.35	29459	60.65	10747	1997.67	31795	237.57	36630	308.18	41661	197.35
940	0.42	329	0.42	291	0.79	243	57.04	267	2.83	201	2.24	380	2.19
70961	66.48	32939	55.62	48079	98.03	11078	1967.32	74503	418.71	94503	521.93	91815	354.8
3352	1.55	202	0.48	235	0.71	171	25.22	214	2.33	249	2.68	262	1.61
20530	21.13	14809	19.71	16851	34.98	8467	1665.95	18969	137	17841	126.37	31380	102.03
1865458	2195.82	782426	1452.83	696019	1635.84	103074	15869.84	392517	2837.28	442777	3246.15	567255	2542.73

Table 7: Wide LB

Opt. val.	**Covering LB**		**Area LB**	
	LB	**Time (s)**	**LB**	**Time (s)**
105	100	0.271037	99	1.25E-07
67	58	0.353536	57	1.23E-07
76	71	0.27374	71	1.27E-07
106	104	0.35181	103	1.35E-07
85	83	0.129346	82	2.35E-07
84	81	0.182686	80	2.79E-07
107	99	0.231393	95	1.32E-07
97	97	0.101953	89	1.22E-07
?	109	0.512909	108	1.24E-07
?	70	0.492993	70	1.58E-07
104	99	0.291328	98	1.46E-07
62	57	0.512773	57	1.29E-07
108	98	0.257797	96	1.22E-07
82	77	0.386635	76	1.05E-07
81	81	0.136615	77	9.70E-08