RC23686 (W0508-022) August 2, 2005
Mathematics

IBM Research Report

An Implementation of a Separation Procedure for Mixed Integer Rounding Inequalities

João P. M. Gonçalves
Lehigh University
Bethlehem, PA 18015

Laszlo Ladanyi
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

An Implementation of a Separation Procedure for Mixed Integer Rounding Inequalities

João P.M. Gonçalves
Lehigh University, Bethlehem, PA 18015

Laszlo Ladanyi
IBM T.J. Watson Research Center, Yorktown Heights, NY 10598

August 12, 2004

Abstract

We describe an implementation of a separation procedure for mixed integer rounding (MIR) inequalities originally proposed by Marchand and Wolsey in 1998. This implementation is included in the Cut Generator Library of the Computational Infrastructure for Operations Research (COIN-OR), which is an initiative of open source software for Operations Research. We detail the three heuristic steps of the separation procedure and provide details of our implementation. We also give useful information for those who want to use it and for those who want to implement other cut separation procedures. Finally, we give results of computational experiments with a standard set of mixed

integer problem instances and show that the performance of our implementation is comparable with other implementations.

1 Introduction

In this report we describe an implementation of a separation procedure for mixed integer rounding (MIR) inequalities. This procedure was proposed by Marchand and Wolsey [4] (for an earlier version of the paper see [3]) and its basic idea is to generate MIR inequalities from constraints or simple aggregations of constraints of the original problem. As explained in [4], the motivation for this idea comes from the observation that several strong valid inequalities based on specific problem structure can be derived as MIR inequalities.

The separation procedure consists of three heuristic steps. The first step is called aggregation and involves combining one or more rows of the constraint matrix in order to obtain a single mixed integer constraint. This is followed by a bound substitution step where each continuous variable is substituted by one of its bounds and associated slack variable. The bound substitution results in a mixed knapsack set that is used in the final step to generate a violated MIR inequality.

The MIR separation procedure was implemented as a C++ class integrated in the cut generation library of the COIN-OR open source initiative (www.coin-or.org). The routine can be inserted in a branch-and-cut system and be useful for solving a variety of mixed integer programming problems.

In order to make this report self-contained, we duplicate in sections 2 and 3 the definition of MIR inequalities and the description of the separation procedure given by Marchand and Wolsey [4]. The

interested reader is encouraged to seek the original source for a detailed account of the theoretical background. In section 4 we describe our implementation of the separation routine. In that section we also give an idea of the steps we took and provide useful information for those who want to use it. In section 5 we describe the computational experiments that we performed and provide an analysis of the results obtained. We also compare those results with those given in [4] and those obtained using ILOG CPLEX. Finally, in section 6 we present our conclusions and give some ideas for future work.

2 Mixed Integer Rounding Inequalities

We start by introducting the Mixed Integer Rounding (MIR) inequality. For details on this inequality the reader is referred to [5, 6]. We consider the single constraint mixed integer set

$$Y = \{(y, x) \in Z_+^{|N|} \times R_+^2 : \sum_{j \in N} a_j y_j + x^+ \leq b + x^-\}.$$

The inequality

$$\sum_{j \in N} \left(\lfloor a_j \rfloor + \frac{(f_j - f)^+}{1 - f} \right) y_j \leq \lfloor b \rfloor + \frac{x^-}{1 - f}, \tag{1}$$

where $f = b - \lfloor b \rfloor$ and $f_j = a_j - \lfloor a_j \rfloor$ for $j \in N$, is valid for Y and is called Mixed Integer Rounding (MIR) inequality.

The inequalities that are used in this work are called complemented MIR (c-MIR) inequalities. In order to derive them we start by defining

a mixed knapsack set as

$$X^{MK} = \left\{ (y,s) \in Z_+^{|N|} \times R_+^1 \colon \sum_{j \in N} a_j y_j \leq b + s, y_j \leq u_j \text{ for } j \in N \right\}.$$

We now let (T,U) be a partition of N and $\delta > 0$. We then complement variables in U and divide the new mixed knapsack constraint by δ, which leads to the following set

$$\left\{ (y,\bar{y},s) \in Z_+^{|T|} \times Z_+^{|U|} \times R_+^1 \colon \sum_{j \in T} \frac{a_j}{\delta} y_j + \sum_{j \in U} \frac{-a_j}{\delta} \bar{y}_j \leq \frac{b - \sum_{j \in U} a_j u_j}{\delta} + \frac{s}{\delta} \right\}.$$

When we generate a MIR inequality valid for this set we get the c-MIR inequality associated with (T,U) and δ. We can write the c-MIR as

$$\sum_{j \in T} G\left(\frac{a_j}{\delta}\right) y_j + \sum_{j \in U} G\left(\frac{-a_j}{\delta}\right) (u_j - y_j) \leq \lfloor \beta \rfloor + \frac{s}{\delta(1-f)},$$

where $\beta = \left(b - \sum_{j \in U} a_j u_j\right)/\delta$, $f = \beta - \lfloor \beta \rfloor$ and $G(d) = \left(\lfloor d \rfloor + (f_d - f)^+ / (1-f)\right)$ with $f_d = d - \lfloor d \rfloor$.

3 The c-MIR Separation Heuristic

We now describe the c-MIR separation heuristic for mixed integer sets that we implemented and that was proposed in [4]. This separation procedure consists of three heuristic steps that can be described as follows:

1. (Aggregation) In this step, a single mixed integer constraint is obtained by combining one or more rows of the problem matrix.
2. (Bound substitution) Here, we construct a mixed knapsack set of the form of X^{MK}. This is done by taking each continuous

variable of the mixed integer constraint derived in the previous step and introducing the slack variable from the corresponding lower or upper bound constraints.

3. (Separation) This step consists of finding a violated c-MIR inequality for the mixed knapsack set X^{MK} created in step 2.

In order to apply the above procedure, we assume that the original problem constraints have been divided in three groups as follows:

(a) The set M of general mixed integer constraints of the form

$$\sum_{j \in P} a_j^i x_j + \sum_{j \in N} g_j^i y_j = b_i \text{ for } i \in M,$$

where x_j for $j \in P$ are real variables and y_j for $j \in N$ are nonnegative integer variables. Note that general mixed integer inequalities are transformed into equalities by adding slack variables and are also included in set M. Also note that Marchand and Wolsey [4] restrict x_j to be real nonnegative variables.

(b) Simple or variable lower and upper bound constraints

$$l_j \leq x_j \leq u_j \quad \text{or} \quad l_j y_j \leq x_j \leq u_j y_j.$$

(c) Other constraints that we ignore.

We also assume that a fractional point (x^*, y^*) is given.

We now detail each of the three heuristic steps. In order to simplify the notation, we assume that all bound constraints are variable lower and upper bounds.

3.1 The Aggregation Heuristic

We suppose that a subset $S \subseteq M$ of the mixed integer rows has been combined to form a single mixed integer constraint

$$\sum_{j \in P} \alpha_j x_j + \sum_{j \in N} \gamma_j y_j = \beta.$$

We also suppose that the bound substitution and the c-MIR separation procedures have been called for this constraint and no violated inequality was found. We try to find another row $r \in M \setminus S$ to combine with the above constraint. In order to do that, we start by constructing the following set

$$P^* = \{k \in P : \alpha_k \neq 0, l_k y_k^* < x_k^* < u_k y_k^*, \text{ and } \exists r \in M \setminus S \text{ with } a_k^r \neq 0\}.$$

If the set P^* is empty, we stop. Otherwise, we choose $k \in P^*$ such that $k = \arg\max\{\Delta_k : k \in P^*\}$ and $\Delta_k = \min\{x_k^* - l_k y_k^*, u_k y_k^* - x_k^*\}$. We select one of the rows $r \in M \setminus S$ with $a_k^r \neq 0$.

After selecting row r, we aggregate it with the current mixed integer constraint. The aggreation is such that the coefficient of x_k becomes zero. This is achieved by doing

$$\begin{aligned}
\alpha_j &\leftarrow \alpha_j - \frac{\alpha_k}{a_k^r} a_j^r \text{ for } j \in P, \\
\gamma_j &\leftarrow \gamma_j - \frac{\alpha_k}{a_k^r} g_j^r \text{ for } j \in N, \\
\beta &\leftarrow \beta - \frac{\alpha_k}{a_k^r} b_r, \\
S &\leftarrow S \cup \{r\}.
\end{aligned}$$

The initialization of the aggregation heuristic consists of selecting a mixed integer row $i \in M$ and setting $S = \{i\}$. The procedure stops

when $|S|$ exceeds the value of MAXAGGR specified by the user.

3.2 The Bound Substitution Heuristic

In this heuristic, we consider the aggregated constraint obtained in the previous step

$$\sum_{j \in P} \alpha_j x_j + \sum_{j \in N} \gamma_j y_j = \beta,$$

as well as the variable bound constraints on the continuous variables x_j for $j \in P$. We then substitute each continuous variable by either letting $x_j = l_j y_j + t_j$ or $x_j = u_j y_j - t_j$, where $t_j \geq 0$. The resulting constraint can be written as

$$\sum_{j \in N} \gamma_j^{'} y_j + \sum_{j \in P} \delta_j^{'} t_j = \beta^{'}.$$

By ignoring the variables t_j with nonnegative coefficients ($\delta_j^{'} \geq 0$) and letting $s = \sum_{j \in P:\, \delta_j^{'} < 0} (-\delta_j^{'}) t_j \geq 0$, we obtain the mixed knapsack set

$$X^{MK} = \left\{ (y, s) \in Z_+^{|N|} \times R_+^1 : \sum_{j \in N} \gamma_j^{'} y_j \leq \beta^{'} + s, y_j \leq u_j \text{ for } j \in N \right\}.$$

In [4], the authors propose three different criteria to select the bound substitution for each continuous variable:

(A) Minimize the difference between each continuous variable and its bound, i.e., if $x_j^* - l_j y_j^* \leq u_j y_j^* - x_j^*$, substitute $x_j = l_j y_j + t_j$. Otherwise, substitute $x_j = u_j y_j - t_j$.

(B) Minimize the value of $s^* = \sum_{j \in P:\, \delta_j^{'} < 0} (\delta_j^{'}) t_j^*$. This is accomplished with the following:

1. If $u_j = \infty$ or $x_j^* = l_j y_j^*$, substitute $x_j = l_j y_j + t_j$.

2. Else if $l_j = -\infty$ or $x_j^* = u_j y_j^*$, substitute $x_j = u_j y_j - t_j$.

3. Else if $\alpha_j < 0$, substitute $x_j = u_j y_j - t_j$.

4. Else if $\alpha_j > 0$, substitute $x_j = l_j y_j + t_j$.

(C) Minimize the value of $\sum_{j \in P:\ \delta_j' > 0} (\delta_j^{'}) t_j^*$. This is obtained by doing:

1. If $u_j = \infty$ or $x_j^* = l_j y_j^*$, substitute $x_j = l_j y_j + t_j$.

2. Else if $l_j = -\infty$ or $x_j^* = u_j y_j^*$, substitute $x_j = u_j y_j - t_j$.

3. Else if $\alpha_j < 0$, substitute $x_j = l_j y_j + t_j$.

4. Else if $\alpha_j > 0$, substitute $x_j = u_j y_j - t_j$.

3.3 The Separation Heuristic

We start by rewriting, with the notation used in section 2, the mixed knapsack set obtained in the bound substitution heuristic

$$X^{MK} = \left\{ (y, s) \in Z_+^{|N|} \times R_+^1 : \sum_{j \in N} a_j y_j \leq b + s, y_j \leq u_j \text{ for } j \in N \right\}.$$

In order to generate a c-MIR inequality, we have to choose the set $U \subseteq N$ and the value of δ. We start by letting $U = \left\{ j \in N : y_j^* \geq \frac{u_j}{2} \right\}$ and $T = N \setminus U$. We define the set $\left\{ a_j : j \in N \text{ and } 0 < y_j^* < u_j \right\}$ as the set of possible values for δ. We then generate a c-MIR inequality for each δ in the previous set and choose the value of $\delta = a_j$ that leads to the c-MIR inequality with the largest violation. We then generate c-MIR inequalities with δ taking the values $\delta = a_j/2, a_j/4, a_j/8$, and choose the value that gives the inequality with the largest violation as our final choice of δ. Taking this value of δ, we try to increase the violation by successively moving each variable in T lying strictly between its bounds to U and generating a new c-MIR inequality. The

variables in T are first ordered by nondecreasing values of $|y_j^* - \frac{u_j}{2}|$.

We run the whole c-MIR separation procedure starting the aggregation heuristic with each mixed integer row in the problem's constraint matrix.

4 Implementation

4.1 Description

The c-MIR separation heuristic was implemented as a C++ class called CglMixedIntegerRounding. This class inherits from the class CglCutGenerator, which is the COIN abstract base class for generating cuts. The class CglMixedIntegerRounding implements the virtual method generateCuts declared in CglCutGenerator. This method has two arguments: 'si' which contains the model data, and 'cs' which contains the collection of cuts generated. We can see 'si' as an input because it contains the model data that is going to be used to generate cuts. On the other hand, we can see 'cs' as output since it is the collection of cuts generated.

The method generateCuts calls the mixIntRoundPreprocess, then reads the problem matrix by row and by column, and finally calls generateMirCuts. The method mixIntRoundPreprocess determines the type of each row, stores the variable upper bound constraints in an array of CglMixIntRoundVUB objects, and stores the variable lower bound constraints in an array of CglMixIntRoundVLB objects.

The method generateMirCuts implements the c-MIR heuristic described in section 3. It basically consists of two loops. The outer loop goes over all mixed integer constraints and the inner loop corresponds to one pass through the c-MIR procedure starting with the constraint

selected in the outer loop.

We have slightly changed the heuristic given in [4]. After the aggregation heuristic, we have a mixed integer constraint that we use to try to generate a violated c-MIR inequality. In our implementation, we not only do that but also multiply the constraint by -1 and try to generate a violated c-MIR inequality from the resulting constraint.

The aggregation heuristic is divided into two methods. We first call SelectRowToAggregate where the set P^* is constructed and a row to use in the aggregation is selected. We then call the method aggregateRow to actually aggregate the row selected with the current mixed integer constraint. The bound substitution and separation heuristics are implemented in the methods boundSubstitution and cMirSeparation, respectively.

4.2 Development steps

In this section, we give a description of the coding process. Our objective is to give interested readers an understanding of the various things that we tried and hopefully provide useful information for someone that would like to improve upon our work or for someone that plans to code other separation procedures.

We started by implementing the separation procedure trying to follow Marchand and Wolsey's paper [4] as close as possible. We implemented the bound substitution criterion A and tested our code not allowing aggregation of rows (MAXAGGR = 1). In order to be able to compare our results with those given in [4], we followed the test procedure described in that paper. Basically, we simulated the root node of a branch and bound tree by consecutively solving the linear programming (LP) relaxation and adding new cuts until the improvement in

the objective function value becomes negligenciable. Comparison with the results given in [4] showed that our code generated less cuts and provided worse bounds than those reported by Marchand and Wolsey.

We then decreased the value of the tolerance used to accept or reject a cut and that helped improve our results although they were still significantly worse than those reported in [4]. At that point, we thought that perhaps there were important implementation details not described in the paper and we went over the heuristic procedures trying to identify what those details could possibly be. One thing that we had noticed was that frequently the bound substitution heuristic failed to return a mixed knapsack constraint. We concluded that that happened when the continuous variable s (see section 3.2) was zero due to the inexistence of coeficients $\delta_j' < 0$ after bound substitution. This observation lead us to modify the bound substitution heuristic such that we can also generate mixed knapsack constraints from the aggregated row multiplied by -1. This increases the chances of finding a violated c-MIR inequality. Testing confirmed that the number of cuts generated increased and the bounds improved.

After the changes described in the previous paragraph, there were still a few instances for which our implemntation performed much worse than Marchand and Wolsey's implementation. We tested again using LP preprocessing but that didn't provide much improvement. We also implemented the other two criteria for bound substitution (B and C) and again there was not significant difference in the results.

It is possible that the results reported in [4] were obtained using integer programming preprocessing which might strongly affect the results. Since integer programming preprocessing is not currently available in COIN-OR, we could not prove that the differences we observe

between the results from our implementation and those from Marchand and Wolsey's implementation are due to that.

The next step was to test the code using aggregation of rows. The way to choose a row to combine with the current aggregated row is not completely detailed in [4]. As described in section 3.1, after we identify the variable $x_k, k \in P^*$ that is furthest from its bounds, we have to select a row $r \in M \setminus S$ with $a_k^r \neq 0$. The paper does not specify which row to choose if there is more than one choice. We implemented two different strategies to select a row to aggregate. The first is to select the row randomly from the set of possible rows. The second is to take the first row that we can find. We also decided to write a version of the code where all possible aggregated rows are generated. This means that for each initial aggregated row, we generate as many new aggregated rows as the number of constraints that can be combined with the original aggregated row. This version of the code turned out to be too slow since the number of aggregated rows can become very large. However, we tested it for some instances allowing up to 3 constraints to be combined in order to form aggregated rows and the bounds do not seem to be much better than by using the other version of the code with any of the two selection criteria described above. We decided to use the strategy of using the first row that we can find in the final version of the code.

4.3 How to use it

The final version of the MIR separation procedure includes three parameters that the user can choose. These parameters are:

MAXAGGR - This is the maximum number of constraints that, in the aggregation heuristic, can be combined to form a single aggregated mixed integer constraint. It takes only positive integer values.

MULTIPLY - This parameter specifies if we also try to generate c-MIR inequalities after multiplying each aggregated constraint by -1. It takes values 'true' or 'false'.

CRITERION - This parameter specifies the criterion used in the bound substitution heuristic. It takes values 1 (criterion A), 2 (criterion B) and 3 (criterion C).

There are two constructors available. The default constructor sets MAXAGGR = 1, MULTIPLY = true, CRITERION = 1. The alternate constructor includes the three parameters above as arguments. There are also public methods available to set and get the values of the three parameters.

5 Experiments

In this section we present results from testing our code on a subset of the mixed integer problem set MIPLIB3.0 [1]. The subset that we used is the same that Marchand and Wolsey [4] used in their computational experiments and contains most of the problems in the set MIPLIB3.0 that have both integer and continuous variables.

Our experiments consisted on simulating the root node of a branch and bound tree. We start by solving the linear programming (LP) relaxation of the original problem and then call the MIR separation routine to generate inequalities that are violated by the current linear programming solution. After adding the new inequalities to the

problem, we solve again the linear programming relaxation and call the MIR separation routine once more. This process is repeated until the percent change of the LP solution in two consecutive iterations is less than 0.01% or the total number of cuts generated exceeds 2000. Our results are compared with those given in [4] and also with those obtained by running the same experiments using ILOG CPLEX 8.1 [2].

We present the results in the form of tables. In table 1, the column "Int Soln" contains the optimal objective function value of the original integer program. The next two columns contain the optimal objective function value of the original LP relaxation ("LP Soln") and the Gap, which is defined as Gap = (Int Soln - LP Soln) / Int Soln x 100. The remaining columns contain the results obtained by Marchand and Wolsey [4]. The first block of three columns corresponds to the case where there is no row aggregation (MAXAGGR = 1) and the second block corresponds to the case where up to 6 rows can be combined to form the aggregated row (MAXAGGR = 6). For each block, we give the number of MIR cuts generated, the LP solution at the root node after adding the cuts, and the gap as defined above. It should be noted that in [4] it is not reported which bound substitution criterion was used and it is also not clear whether preprocessing was used or not.

The results given in the other tables of this report are presented in blocks of three columns as those above. Tables 2 and 3 contain the results obtained with ILOG CPLEX 8.1. These results were obtained by simulating the root node without using preprocessing. Table 2 contains the case where there is no row aggregation (MAXAGGR = 1) and the cases where the maximum number of rows that form the aggregated row is 2 (MAXAGGR = 2) and 3 (MAXAGGR = 3). Table 3 contains the cases with MAXAGGR = 4, 5, and 6.

Tables 4 - 15 contain the results obtained with our implementation of the MIR separation procedure. Table 4 corresponds to the case where we don't allow aggregation to happen, use the bound substitution criterion A and do not use preprocessing. The first block is for the case where besides using the aggregated row to generate MIR cuts, we also use the aggregated row multiplied by -1. The second block is for the case where we only use the original aggregated constraints. Table 5 corresponds to the case with the same parameters as in the previous table except that in this case we used preprocessing. The same tables were generated for the results using the bound substitution criterions B (tables 6 and 7) and C (tables 8 and 9). Tables 10 and 11 contain the results for the cases where up to 2, 4 and 6 rows can be combined to form the aggregated constraint. For both tables, the bound substitution criterion A was used and preprocessing was included. Also, besides using the aggregated row to generate MIR cuts, we also use the aggregated row multiplied by -1. In table 10, each row that is used in the aggregation process is selected randomly from the set of possible rows. In table 11, the first row found is used in the aggregation process. Tables similar to the previous two were created for the cases where the bound substitution criterion is B (tables 12 and 13) and C (tables 14 and 15).

The last row of each table contains the average number of cuts generated and the average gap. The averages were calculated across all instances except "noswot", "pk1", and "qiu". The results for these three instances were not included in the calculation of the averages because the LP objective function values never changed for all tests performed.

We start by analyzing the results from our implementation of the

MIR separation procedure when aggregation of rows is not allowed (MAXAGGR = 1). If we compare the averages from tables 4 to 9, we can conclude that using the bound substitution criterion A or B produces better bounds than using criterion C. However, the number of cuts generated is larger for the former two criteria. This trade-off between the bounds and the number of cuts generated is observed for most of the results given here, including the results reported in [4] and those obtained with ILOG CPLEX.

From tables 4 to 9 we can also conclude that the bounds improve when, in addition to using the original constraints, we also use the problem constraints multiplied by -1 to generate MIR cuts. Moreover, the use of LP preprocessing has in general a small impact on the bounds and can result in better or worse bounds.

Comparing the results from our implementation with those given in tables 1 and 2 for MAXAGGR = 1, we can see that the best of our average bounds (8.040) is worse than the average bound reported by Marchand and Wolsey [4] (5.956) but much better than the average bound obtained with ILOG CPLEX (14.482). Looking at the results of each instance corresponding to the best average bound obtained with our implementation (first block in table 5) and reported in [4], we verify that the bounds from both implementations are very close except for 3 instances. If we remove those instances from the averages, we get a value of 6.339 for the average gap in our implementation and 6.110 for the Marchand and Wolsey implementation.

We now look at the results when we allow the aggregation of rows. We tested two ways of selecting the next row to use in the aggregation process. The first way we used is to select a row randomly from the set of rows that can be selected. The second strategy is to take the

first row that we can find. The results given in tables 10 to 15 indicate that the two strategies perform similarly.

Looking at the influence of the bound substitution criterion used, we can conclude that, in this case, criterion A provides better bounds on average than criterion B. As verified for the case with MAXAGGR = 1, criterion C gives the worst bounds.

Allowing more constraints to be combined in order to form the aggregated constraint has the effect of improving the average gap. However, the incremental improvement decreases with the increase of MAXAGGR and after a certain point the average gap increases again. In some cases, the average gap increases when we change MAXAGGR from 4 to 6. This agrees with the findings of Marchand and Wolsey [4] who reported that there is no improvement for values of MAXAGGR greater than 6. The results obtained using ILOG CPLEX show an improvement up to MAXAGGR = 6. We have not tested for values of MAXAGGR greater than 6.

Comparison of the three implementations using MAXAGGR = 6 shows that the best average gap from our implementation (4.295, see table 11) falls between the average gap obtained by Marchand and Wolsey (2.367, see table 1) and that obtained using ILOG CPLEX (7.707, see table 3).

6 Conclusions

In this report, we have described our implementation of the mixed integer rounding separation procedure proposed by Marchand and Wolsey [4]. This implementation is part of the COIN-OR cut library.

We have included some notes regarding the implementation process

that will hopefully help the user quickly understand the important details and also provide some insights regarding the problems associated with the implementation of a cut generator. We describe a strategy to help increase the number of c-MIR inequalities generated that is not given in [4] and that has proven to be successful. Also, we give a description of the parameters that users can use to try to improve the performance of their particular applications.

We provide results from computational experiments. We compare our results with those reported by Marchand and Wolsey and also those obtained with ILOG CPLEX 8.1. We concluded that our implementation is competitive with those in terms of number of cuts generated and bounds provided. We have not compared times because they are not available in [4].

Regarding the selection of rows to use in the aggregation process we decided to implement a simple strategy that consists of selecting the first row that we can find. This strategy seems to perform well when compared to the case where we generate all possible aggregated rows.

The current implementation does not use constraints that contain free variables. This possibility should be included in the future. Also, we always use the inital matrix (without any cuts added) to generate c-MIR inequalities. We assume that these constraints remain at the beginning of the matrix and are never removed from the problem. Each time the c-MIR separation procedure is called, we read in those rows and proceed. This might need to be changed in case there is a modification of the original matrix that is not compatible with our assumption.

7 Acknowledgments

The work presented in this report was done during the summer 2004 when the first author was visiting the Mathematical Sciences Department of the IBM T.J. Watson Research Center. We are thankful to Dr. Oktay Gunlunk for his insightful comments during several very productive discussions.

References

[1] R. E. Bixby, S. Ceria, C. M. McZeal, and M. W. P. Savelsbergh. An updated mixed integer programming library: MIPLIB 3, 1998. Text and problems available at http://www.caam.rice.edu/ bixby/miplib/miplib3.html.

[2] ILOG, Inc., Mountain View, CA. *ILOG CPLEX 8.1 User's Manual*.

[3] H. Marchand and L. A. Wolsey. Aggregation and Mixed Integer Rounding to solve MIPs. CORE Discussion Paper 9839, CORE, Université Catholique de Louvain, 1998.

[4] H. Marchand and L. A. Wolsey. Aggregation and Mixed Integer Rounding to solve MIPs. *Operations Research*, 49(3):363–371, 2001.

[5] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, 1988.

[6] L. A. Wolsey. *Integer Programming*. John Wiley & Sons, 1998.

		Original LP relaxation		MAXAGGR = 1			MAXAGGR = 6		
Instance	Int Soln	LP Soln	Gap	Cuts	LP Soln	Gap	Cuts	LP Soln	Gap
arki001	7580813.05	7579599.81	0.016	20	7579735.00	0.014	176	7579798.00	0.013
bell3a	878430.32	862578.64	1.805	6	869221.00	1.048	13	873351.00	0.578
bell5	8966406.49	8608417.95	3.993	6	8609853.00	3.977	18	8621775.00	3.844
blend2	7.60	6.92	8.992	110	7.08	6.830	693	7.17	5.647
danoint	65.67	62.64	4.618	87	62.67	4.568	863	62.72	4.492
dcmulti	188182.00	183975.54	2.235	56	184275.00	2.076	64	184283.00	2.072
egout	568.10	149.59	73.669	30	539.00	5.123	213	561.00	1.250
fixnet6	3983.00	1200.88	69.850	203	3624.00	9.013	1788	3832.00	3.791
flugpl	1201500.00	1167185.73	2.856	1	1167875.00	2.799	1	1167875.00	2.799
gen	112313.00	112130.00	0.163	42	112313.00	0.000	42	112313.00	0.000
gesa2	25779856.37	25476489.68	1.177	469	25727081.00	0.205	868	25778425.00	0.006
gesa2o	25779856.37	25476489.68	1.177	197	25592240.00	0.728	662	25775806.00	0.016
gesa3	27991042.65	27833632.45	0.562	166	27930251.00	0.217	331	27955301.00	0.128
gesa3o	27991042.65	27833632.45	0.562	102	27919041.00	0.257	240	27943244.00	0.171
khb05250	106940226.00	95919464.00	10.306	174	106747470.00	0.180	259	106857080.00	0.078
misc06	12850.86	12841.69	0.071	0	12841.00	0.077	0	12841.00	0.077
mod011	-54558535.00	-62121982.55	13.863	100	-61525216.00	12.769	922	-58449690.00	7.132
modglob	20740508.00	20430947.00	1.493	263	20681346.00	0.285	530	20726103.00	0.069
noswot	-43.00	-43.00	0.000	16	-43.00	0.000	335	-43.00	0.000
pk1	11.00	0.00	100.000	9	0.00	100.000	9	0.00	100.000
pp08a	7350.00	2748.35	62.608	131	5565.00	24.286	593	7123.00	3.088
pp08aCUTS	7350.00	5480.61	25.434	23	5565.00	24.286	1071	7219.00	1.782
qiu	-132.87	-931.64	601.149	0	-931.64	601.149	0	-931.64	601.149
rentacar	30356761.00	28806137.64	5.108	98	29395235.00	3.167	175	29449924.00	2.987
rgn	82.20	48.80	40.633	73	68.00	17.275	151	82.20	0.000
rout	1077.56	981.86	8.881	895	982.17	8.852	1444	982.64	8.809
set1ch	54537.75	32007.73	41.311	466	48220.00	11.584	1328	51814.00	4.994
vpm1	20.00	15.42	22.917	149	19.50	2.500	468	20.00	0.000
vpm2	13.75	9.89	28.078	166	12.00	12.727	652	12.69	7.709
Average			16.630	155		5.956	522		2.367

Table 1: Integer solution, original LP relaxation, and MIR results obtained by Marchand and Wolsey [4].

	MAXAGGR = 1			MAXAGGR = 2			MAXAGGR = 3		
Instance	Cuts	LP Soln	Gap	Cuts	LP Soln	Gap	Cuts	LP Soln	Gap
arki001	4	7579689.59	0.015	23	7579707.77	0.015	50	7579814.90	0.013
bell3a	0	862578.64	1.805	12	872494.79	0.676	13	873199.39	0.595
bell5	0	8608417.95	3.993	8	8618834.03	3.876	10	8619044.34	3.874
blend2	0	6.92	8.992	20	7.03	7.492	20	7.03	7.492
danoint	0	62.64	4.618	37	65.66	0.018	59	62.68	4.551
dcmulti	0	183975.54	2.235	16	184102.63	2.168	25	184455.91	1.980
egout	0	149.59	73.669	47	287.31	49.427	59	312.93	44.917
fixnet6	0	1200.88	69.850	113	1585.14	60.202	226	1823.74	54.212
flugpl	0	1167185.73	2.856	1	1167875.17	2.799	1	1167875.17	2.799
gen	0	112130.00	0.163	28	112196.83	0.103	28	112196.83	0.103
gesa2	63	25683253.55	0.375	83	25696011.23	0.325	84	25697100.59	0.321
gesa2o	0	25476489.68	1.177	67	25576317.33	0.790	73	25582081.96	0.767
gesa3	40	27907201.80	0.300	58	27914338.06	0.274	58	27916636.02	0.266
gesa3o	0	27833632.45	0.562	32	27897305.51	0.335	41	27900351.40	0.324
khb05250	0	95919464.00	10.306	91	106294351.20	0.604	116	106715041.48	0.211
misc06	0	12841.69	0.071	0	12841.69	0.071	0	12841.69	0.071
mod011	0	-62121982.55	13.863	0	-62121982.55	13.863	0	-62121982.55	13.863
modglob	0	20430947.00	1.493	105	20552415.01	0.907	89	20584557.41	0.752
noswot	15	-43.00	0.000	19	-43.00	0.000	21	-43.00	0.000
pk1	0	0.00	100.000	0	0.00	100.000	0	0.00	100.000
pp08a	0	2748.35	62.608	118	5659.46	23.000	87	5264.57	28.373
pp08aCUTS	3	5488.31	25.329	101	6451.84	12.220	119	6827.09	7.114
qiu	0	-931.64	601.149	0	-931.64	601.149	0	-931.64	601.149
rentacar	0	28806137.64	5.108	2	29588163.24	2.532	2	29588163.24	2.532
rgn	42	68.00	17.275	60	68.36	16.841	59	69.12	15.907
rout	2	981.86	8.881	9	981.86	8.881	9	981.86	8.881
set1ch	0	32007.73	41.311	231	38326.34	29.725	175	42813.58	21.497
vpm1	55	19.00	5.000	65	19.00	5.000	63	19.50	2.500
vpm2	81	11.73	14.678	128	12.16	11.529	142	12.28	10.704
Average	11		14.482	56		9.757	62		9.024

Table 2: MIR results obtained with ILOG CPLEX 8.1 [2].

	MAXAGGR = 4			MAXAGGR = 5			MAXAGGR = 6		
Instance	Cuts	LP Soln	Gap	Cuts	LP Soln	Gap	Cuts	LP Soln	Gap
arki001	51	7579814.90	0.013	49	7579814.90	0.013	49	7579814.90	0.013
bell3a	12	873474.77	0.564	12	873474.77	0.564	12	873474.77	0.564
bell5	11	8619730.84	3.866	11	8619730.84	3.866	11	8619730.84	3.866
blend2	20	7.03	7.492	20	7.03	7.492	20	7.03	7.492
danoint	61	62.69	4.540	74	62.70	4.527	97	62.70	4.520
dcmulti	65	185631.10	1.356	77	185901.24	1.212	80	185888.87	1.219
egout	68	321.05	43.487	66	325.71	42.667	77	343.35	39.562
fixnet6	227	1869.42	53.065	233	1908.00	52.096	278	1964.35	50.682
flugpl	1	1167875.17	2.799	1	1167875.17	2.799	1	1167875.17	2.799
gen	28	112196.83	0.103	29	112198.14	0.102	29	112198.14	0.102
gesa2	84	25697044.79	0.321	96	25701710.14	0.303	94	25703773.84	0.295
gesa2o	76	25592437.39	0.727	84	25627357.79	0.592	105	25656392.10	0.479
gesa3	67	27917685.98	0.262	91	27937765.01	0.190	93	27938994.81	0.186
gesa3o	50	27905998.79	0.304	57	27913966.79	0.275	62	27916295.55	0.267
khb05250	118	106735952.05	0.191	111	106863649.78	0.072	116	106860376.48	0.075
misc06	7	12846.51	0.034	11	12848.78	0.016	12	12848.78	0.016
mod011	35	-61371417.56	12.487	91	-59525120.22	9.103	113	-58975851.31	8.096
modglob	93	20594929.72	0.702	92	20602177.62	0.667	90	20599751.68	0.679
noswot	21	-43.00	0.000	21	-43.00	0.000	21	-43.00	0.000
pk1	0	0.00	100.000	0	0.00	100.000	0	0.00	100.000
pp08a	119	6006.88	18.274	119	6025.25	18.024	117	6030.61	17.951
pp08aCUTS	123	7101.65	3.379	128	7177.35	2.349	122	7178.25	2.337
qiu	0	-931.64	601.149	0	-931.64	601.149	0	-931.64	601.149
rentacar	6	29588163.24	2.532	6	29588163.24	2.532	6	29588163.24	2.532
rgn	50	69.09	15.954	50	70.40	14.351	50	70.40	14.351
rout	9	981.86	8.881	9	981.86	8.881	9	981.86	8.881
set1ch	178	42909.15	21.322	164	41304.64	24.264	204	43433.66	20.360
vpm1	58	19.50	2.500	58	19.50	2.500	58	19.50	2.500
vpm2	141	12.30	10.571	141	12.30	10.571	141	12.30	10.571
Average	68		8.297	72		8.078	79		7.707

Table 3: MIR results obtained with ILOG CPLEX 8.1 [2].

	MULTIPLY = TRUE			MULTIPLY = FALSE		
Instance	Cuts	LP Soln	Gap	Cuts	LP Soln	Gap
arki001	8	7579685.10	0.015	8	7579685.10	0.015
bell3a	4	865207.82	1.505	0	862578.64	1.805
bell5	5	8609681.32	3.978	0	8608417.95	3.993
blend2	33	7.03	7.469	30	7.03	7.469
danoint	15	62.65	4.598	12	62.65	4.598
dcmulti	18	184097.64	2.170	0	183975.54	2.235
egout	66	433.86	23.630	54	433.86	23.630
fixnet6	67	3192.04	19.858	0	1200.88	69.850
flugpl	0	1167185.73	2.856	0	1167185.73	2.856
gen	37	112312.60	0.000	37	112312.60	0.000
gesa2	219	25698221.44	0.317	189	25695509.00	0.327
gesa2o	162	25578329.50	0.782	127	25573264.48	0.801
gesa3	83	27916057.74	0.268	83	27916057.74	0.268
gesa3o	66	27903817.53	0.312	66	27903817.53	0.312
khb05250	0	95919464.00	10.306	0	95919464.00	10.306
misc06	0	12841.69	0.071	0	12841.69	0.071
mod011	0	-62121982.55	13.863	0	-62121982.55	13.863
modglob	2	20431180.77	1.491	0	20430947.62	1.493
noswot	41	-43.00	0.000	12	-43.00	0.000
pk1	0	0.00	100.000	0	0.00	100.000
pp08a	5	5489.24	25.317	5	5489.24	25.317
pp08aCUTS	64	5486.59	25.353	64	5486.59	25.353
qiu	0	-931.64	601.149	0	-931.64	601.149
rentacar	1	28806137.64	5.108	0	28806137.64	5.108
rgn	31	68.00	17.275	0	48.80	40.633
rout	15	981.86	8.881	11	981.86	8.881
set1ch	146	40681.07	25.408	146	40681.07	25.408
vpm1	37	19.50	2.500	37	19.50	2.500
vpm2	114	11.80	14.165	114	11.80	14.165
Average	46		8.365	38		11.202
Average	46		8.365	38		11.202

Table 4: Our MIR results (CRITERION = A, MAXAGGR = 1, Preprocessing = False).

	MULTIPLY = TRUE			MULTIPLY = FALSE		
Instance	Cuts	LP Soln	Gap	Cuts	LP Soln	Gap
arki001	8	7579685.10	0.015	8	7579685.10	0.015
bell3a	5	865600.11	1.461	0	862578.64	1.805
bell5	5	8609681.32	3.978	0	8608417.95	3.993
blend2	32	7.03	7.469	29	7.03	7.469
danoint	15	62.65	4.598	12	62.65	4.598
dcmulti	0	183975.54	2.235	0	183975.54	2.235
egout	74	377.48	33.555	36	219.61	61.344
fixnet6	73	3192.06	19.858	1	1200.89	69.850
flugpl	0	1167185.73	2.856	0	1167185.73	2.856
gen	74	112295.09	0.016	74	112295.09	0.016
gesa2	231	25697897.26	0.318	192	25695086.11	0.329
gesa2o	402	25575575.18	0.792	30	25477050.40	1.175
gesa3	93	27919041.40	0.257	87	27919041.40	0.257
gesa3o	157	27917741.02	0.262	4	27834329.24	0.560
khb05250	2	96437115.00	9.821	1	96214626.75	10.030
misc06	0	12841.69	0.071	0	12841.69	0.071
mod011	5	-62040567.51	13.714	0	-62121982.55	13.863
modglob	2	20431180.77	1.491	0	20430947.62	1.493
noswot	41	-43.00	0.000	12	-43.00	0.000
pk1	0	0.00	100.000	0	0.00	100.000
pp08a	19	5648.20	23.154	16	5648.20	23.154
pp08aCUTS	70	5561.36	24.335	70	5561.36	24.335
qiu	0	-931.64	601.149	0	-931.64	601.149
rentacar	5	29036336.63	4.350	0	28806137.64	5.108
rgn	31	68.00	17.275	0	48.80	40.633
rout	13	981.86	8.881	9	981.86	8.881
set1ch	163	48219.59	11.585	158	45374.74	16.801
vpm1	32	19.50	2.500	32	19.50	2.500
vpm2	121	11.80	14.191	121	11.80	14.191
Average	63		8.040	34		12.214

Table 5: Our MIR results (CRITERION = A, MAXAGGR = 1, Preprocessing = True).

	MULTIPLY = TRUE			MULTIPLY = FALSE		
Instance	Cuts	LP Soln	Gap	Cuts	LP Soln	Gap
arki001	0	7579599.81	0.016	0	7579599.81	0.016
bell3a	4	865207.82	1.505	0	862578.64	1.805
bell5	5	8609681.32	3.978	0	8608417.95	3.993
blend2	42	7.04	7.339	37	7.04	7.339
danoint	40	62.66	4.585	40	62.66	4.585
dcmulti	6	184062.21	2.189	0	183975.54	2.235
egout	46	452.54	20.342	42	452.54	20.342
fixnet6	134	3192.04	19.858	0	1200.88	69.850
flugpl	0	1167185.73	2.856	0	1167185.73	2.856
gen	37	112312.60	0.000	37	112312.60	0.000
gesa2	151	25693920.53	0.333	150	25693920.53	0.333
gesa2o	80	25581809.26	0.768	68	25575727.66	0.792
gesa3	76	27914941.33	0.272	76	27914941.33	0.272
gesa3o	41	27909808.87	0.290	41	27909808.87	0.290
khb05250	0	95919464.00	10.306	0	95919464.00	10.306
misc06	0	12841.69	0.071	0	12841.69	0.071
mod011	0	-62121982.55	13.863	0	-62121982.55	13.863
modglob	1	20431051.50	1.492	0	20430947.62	1.493
noswot	41	-43.00	0.000	12	-43.00	0.000
pk1	0	0.00	100.000	0	0.00	100.000
pp08a	6	5489.24	25.317	6	5489.24	25.317
pp08aCUTS	64	5486.56	25.353	64	5486.56	25.353
qiu	0	-931.64	601.149	0	-931.64	601.149
rentacar	1	28806137.64	5.108	0	28806137.64	5.108
rgn	25	64.60	21.411	0	48.80	40.633
rout	13	981.86	8.881	11	981.86	8.881
set1ch	144	40660.88	25.445	144	40660.88	25.445
vpm1	78	19.50	2.500	78	19.50	2.500
vpm2	86	11.80	14.166	86	11.80	14.166
Average	42		8.394	34		11.071

Table 6: Our MIR results (CRITERION = B, MAXAGGR = 1, Preprocessing = False).

	MULTIPLY = TRUE			MULTIPLY = FALSE		
Instance	Cuts	LP Soln	Gap	Cuts	LP Soln	Gap
arki001	0	7579599.81	0.016	0	7579599.81	0.016
bell3a	5	865600.11	1.461	0	862578.64	1.805
bell5	5	8609681.32	3.978	0	8608417.95	3.993
blend2	46	7.05	7.232	42	7.05	7.232
danoint	41	62.66	4.585	40	62.66	4.585
dcmulti	0	183975.54	2.235	0	183975.54	2.235
egout	57	368.59	35.118	33	223.74	60.616
fixnet6	143	3194.76	19.790	0	1200.88	69.850
flugpl	0	1167185.73	2.856	0	1167185.73	2.856
gen	78	112304.11	0.008	78	112304.11	0.008
gesa2	153	25693927.16	0.333	151	25693927.16	0.333
gesa2o	142	25583263.32	0.763	2	25476510.46	1.177
gesa3	78	27916057.74	0.268	78	27916057.74	0.268
gesa3o	92	27910855.17	0.286	2	27833985.75	0.561
khb05250	4	96437115.00	9.821	2	96214626.75	10.030
misc06	0	12841.69	0.071	0	12841.69	0.071
mod011	0	-62121982.55	13.863	0	-62121982.55	13.863
modglob	1	20431051.50	1.492	0	20430947.62	1.493
noswot	41	-43.00	0.000	12	-43.00	0.000
pk1	0	0.00	100.000	0	0.00	100.000
pp08a	17	5644.32	23.207	17	5646.86	23.172
pp08aCUTS	70	5561.15	24.338	70	5561.15	24.338
qiu	0	-931.64	601.149	0	-931.64	601.149
rentacar	5	29036336.63	4.350	0	28806137.64	5.108
rgn	25	64.60	21.411	0	48.80	40.633
rout	11	981.86	8.881	9	981.86	8.881
set1ch	231	48131.29	11.747	227	45287.48	16.961
vpm1	89	19.50	2.500	89	19.50	2.500
vpm2	100	11.79	14.225	100	11.79	14.225
Average	54		8.263	36		12.185

Table 7: Our MIR results (CRITERION = B, MAXAGGR = 1, Preprocessing = True).

	MULTIPLY = TRUE			MULTIPLY = FALSE		
Instance	Cuts	LP Soln	Gap	Cuts	LP Soln	Gap
arki001	6	7579673.59	0.015	6	7579673.59	0.015
bell3a	2	864776.39	1.554	0	862578.64	1.805
bell5	4	8609681.32	3.978	0	8608417.95	3.993
blend2	0	6.92	8.992	0	6.92	8.992
danoint	2	62.64	4.614	2	62.64	4.614
dcmulti	1	183984.74	2.230	0	183975.54	2.235
egout	30	297.61	47.613	23	297.56	47.621
fixnet6	96	2102.24	47.220	0	1200.88	69.850
flugpl	0	1167185.73	2.856	0	1167185.73	2.856
gen	37	112312.60	0.000	37	112312.60	0.000
gesa2	132	25691418.02	0.343	121	25691007.71	0.345
gesa2o	36	25519933.22	1.008	25	25503932.66	1.070
gesa3	49	27911256.94	0.285	49	27911256.94	0.285
gesa3o	19	27869165.09	0.435	19	27869165.09	0.435
khb05250	0	95919464.00	10.306	0	95919464.00	10.306
misc06	0	12841.69	0.071	0	12841.69	0.071
mod011	0	-62121982.55	13.863	0	-62121982.55	13.863
modglob	1	20431051.50	1.492	0	20430947.62	1.493
noswot	41	-43.00	0.000	12	-43.00	0.000
pk1	0	0.00	100.000	0	0.00	100.000
pp08a	0	5480.61	25.434	0	5480.61	25.434
pp08aCUTS	35	4519.98	38.504	35	4519.98	38.504
qiu	0	-931.64	601.149	0	-931.64	601.149
rentacar	1	28806137.64	5.108	0	28806137.64	5.108
rgn	21	62.00	24.574	0	48.80	40.633
rout	4	981.86	8.881	4	981.86	8.881
set1ch	122	40044.52	26.575	122	40044.52	26.575
vpm1	36	16.96	15.215	36	16.96	15.215
vpm2	56	10.86	21.054	56	10.86	21.054
Average	27		12.009	21		13.510

Table 8: Our MIR results (CRITERION = C, MAXAGGR = 1, Preprocessing = False).

	MULTIPLY = TRUE			MULTIPLY = FALSE		
Instance	Cuts	LP Soln	Gap	Cuts	LP Soln	Gap
arki001	6	7579683.73	0.015	6	7579683.73	0.015
bell3a	3	865168.68	1.510	0	862578.64	1.805
bell5	4	8609681.32	3.978	0	8608417.95	3.993
blend2	0	6.92	8.992	0	6.92	8.992
danoint	0	62.64	4.618	0	62.64	4.618
dcmulti	0	183975.54	2.235	0	183975.54	2.235
egout	31	255.61	55.007	11	180.72	68.188
fixnet6	98	2063.77	48.186	1	1201.14	69.843
flugpl	0	1167185.73	2.856	0	1167185.73	2.856
gen	55	112300.13	0.011	55	112300.13	0.011
gesa2	132	25691715.24	0.342	118	25691007.71	0.345
gesa2o	68	25514882.72	1.028	2	25478262.95	1.170
gesa3	48	27911748.43	0.283	48	27911748.43	0.283
gesa3o	36	27861421.71	0.463	0	27833632.45	0.562
khb05250	0	95919464.00	10.306	0	95919464.00	10.306
misc06	0	12841.69	0.071	0	12841.69	0.071
mod011	13	-62004268.35	13.647	0	-62121982.55	13.863
modglob	1	20431051.50	1.492	0	20430947.62	1.493
noswot	41	-43.00	0.000	12	-43.00	0.000
pk1	0	0.00	100.000	0	0.00	100.000
pp08a	0	5480.61	25.434	0	5480.61	25.434
pp08aCUTS	35	4321.50	41.204	35	4321.50	41.204
qiu	0	-931.64	601.149	0	-931.64	601.149
rentacar	5	29036336.63	4.350	0	28806137.64	5.108
rgn	21	62.00	24.574	0	48.80	40.633
rout	6	981.86	8.881	4	981.86	8.881
set1ch	140	40910.86	24.986	135	37996.04	30.331
vpm1	31	16.67	16.654	31	16.67	16.654
vpm2	54	10.93	20.539	54	10.93	20.539
Average	30		12.372	19		14.594

Table 9: Our MIR results (CRITERION = C, MAXAGGR = 1, Preprocessing = True).

	MAXAGGR = 2			MAXAGGR = 4			MAXAGGR = 6		
Instance	Cuts	LP Soln	Gap	Cuts	LP Soln	Gap	Cuts	LP Soln	Gap
arki001	27	7579751.41	0.014	52	7579731.12	0.014	78	7579747.99	0.014
bell3a	9	869442.13	1.023	14	869565.25	1.009	14	869565.25	1.009
bell5	9	8610266.93	3.972	14	8610266.93	3.972	12	8610266.93	3.972
blend2	92	7.08	6.879	201	7.10	6.527	185	7.10	6.552
danoint	51	62.67	4.571	124	62.68	4.550	183	62.68	4.560
dcmulti	12	184117.72	2.160	97	184821.96	1.786	172	185263.01	1.551
egout	166	427.41	24.765	346	430.14	24.284	470	429.11	24.466
fixnet6	1326	3242.68	18.587	2022	3210.66	19.391	2432	3206.33	19.500
flugpl	0	1167185.73	2.856	0	1167185.73	2.856	0	1167185.73	2.856
gen	74	112295.09	0.016	74	112295.09	0.016	74	112295.09	0.016
gesa2	352	25704881.34	0.291	462	25726848.75	0.206	462	25726848.75	0.206
gesa2o	659	25623430.12	0.607	1032	25693185.26	0.336	1134	25708466.35	0.277
gesa3	168	27929977.10	0.218	405	27952148.42	0.139	436	27952231.59	0.139
gesa3o	230	27925726.92	0.233	451	27950042.46	0.146	417	27950887.81	0.143
khb05250	64	105874167.26	0.997	141	106590925.56	0.327	186	106817454.87	0.115
misc06	0	12841.69	0.071	0	12841.69	0.071	0	12841.69	0.071
mod011	10	-61968347.73	13.581	87	-61334241.19	12.419	74	-61713022.53	13.113
modglob	217	20582301.83	0.763	835	20613240.05	0.614	1485	20608705.89	0.635
noswot	41	-43.00	0.000	41	-43.00	0.000	41	-43.00	0.000
pk1	0	0.00	100.000	0	0.00	100.000	0	0.00	100.000
pp08a	159	6734.39	8.376	520	7186.33	2.227	700	7221.74	1.745
pp08aCUTS	194	6706.69	8.753	492	7134.50	2.932	855	7156.70	2.630
qiu	0	-931.64	601.149	0	-931.64	601.149	0	-931.64	601.149
rentacar	21	29036336.63	4.350	126	29281038.31	3.544	128	29108564.57	4.112
rgn	69	73.50	10.584	114	81.80	0.486	114	81.80	0.486
rout	19	981.86	8.881	19	981.86	8.881	19	981.86	8.881
set1ch	363	51005.05	6.478	873	51784.12	5.049	1424	51899.81	4.837
vpm1	59	19.08	4.583	104	20.00	0.000	109	20.00	0.000
vpm2	400	12.30	10.550	816	12.52	8.935	835	12.31	10.455
Average	183		5.545	362		4.258	461		4.321

Table 10: Our MIR results (CRITERION = A, MULTIPLY = TRUE, Preprocessing = True, Row to aggregate: select randomly).

	MAXAGGR = 2			MAXAGGR = 4			MAXAGGR = 6		
Instance	Cuts	LP Soln	Gap	Cuts	LP Soln	Gap	Cuts	LP Soln	Gap
arki001	14	7579685.10	0.015	29	7579685.10	0.015	59	7579685.10	0.015
bell3a	11	869442.13	1.023	15	869660.65	0.998	15	869660.65	0.998
bell5	11	8610266.93	3.972	26	8616747.65	3.900	37	8618739.80	3.877
blend2	171	7.11	6.484	348	7.11	6.474	336	7.08	6.766
danoint	104	62.69	4.544	556	62.70	4.525	860	62.70	4.523
dcmulti	18	184260.07	2.084	40	184260.07	2.084	77	184260.49	2.084
egout	158	406.82	28.389	332	412.68	27.358	534	414.32	27.069
fixnet6	1326	3242.68	18.587	2022	3210.66	19.391	2432	3206.33	19.500
flugpl	0	1167185.73	2.856	0	1167185.73	2.856	0	1167185.73	2.856
gen	74	112295.09	0.016	74	112295.09	0.016	74	112295.09	0.016
gesa2	352	25704881.34	0.291	462	25726848.75	0.206	462	25726848.75	0.206
Gesa2o	710	25619689.06	0.621	1233	25689303.27	0.351	1518	25700728.69	0.307
gesa3	168	27929977.10	0.218	370	27942630.56	0.173	390	27942713.74	0.173
Gesa3o	243	27925967.96	0.232	462	27950378.34	0.145	500	27950889.77	0.143
khb05250	63	105874167.26	0.997	128	106571562.52	0.345	192	106816948.02	0.115
misc06	0	12841.69	0.071	0	12841.69	0.071	0	12841.69	0.071
mod011	19	-61751232.74	13.183	127	-60673198.83	11.208	204	-59548057.34	9.145
modglob	217	20582301.83	0.763	835	20613240.05	0.614	1485	20608705.89	0.635
noswot	41	-43.00	0.000	41	-43.00	0.000	41	-43.00	0.000
pk1	0	0.00	100.000	0	0.00	100.000	0	0.00	100.000
pp08a	194	6706.69	8.753	618	7197.96	2.069	1235	7238.04	1.523
pp08aCUTS	148	6756.69	8.072	492	7134.50	2.932	855	7156.70	2.630
qiu	0	-931.64	601.149	0	-931.64	601.149	0	-931.64	601.149
rentacar	11	29036336.63	4.350	27	29036336.63	4.350	40	29036336.63	4.350
rgn	69	73.50	10.584	114	81.80	0.486	114	81.80	0.486
rout	19	981.86	8.881	19	981.86	8.881	19	981.86	8.881
set1ch	363	51005.05	6.478	873	51784.12	5.049	1424	51899.81	4.837
vpm1	59	19.08	4.583	104	20.00	0.000	109	20.00	0.000
vpm2	400	12.30	10.550	816	12.52	8.935	835	12.31	10.455
Average	189		5.638	389		4.363	531		4.295

Table 11: Our MIR results (CRITERION = A, MULTIPLY = TRUE, Preprocessing = True, Row to aggregate: select first found).

	MAXAGGR = 2			MAXAGGR = 4			MAXAGGR = 6		
Instance	Cuts	LP Soln	Gap	Cuts	LP Soln	Gap	Cuts	LP Soln	Gap
arki001	13	7579674.92	0.015	25	7579712.99	0.015	30	7579727.92	0.014
bell3a	13	869478.49	1.019	27	869734.35	0.990	29	869770.09	0.986
bell5	9	8610266.93	3.972	14	8610266.93	3.972	12	8610266.93	3.972
blend2	69	7.06	7.047	231	7.11	6.450	153	7.08	6.855
danoint	58	62.67	4.566	74	62.67	4.562	96	62.67	4.564
dcmulti	13	184117.75	2.160	67	184415.30	2.002	217	185507.39	1.421
egout	100	401.77	29.278	143	392.61	30.891	187	407.69	28.236
fixnet6	635	3495.17	12.248	1849	3553.43	10.785	2069	3457.45	13.195
flugpl	0	1167185.73	2.856	0	1167185.73	2.856	0	1167185.73	2.856
gen	78	112304.11	0.008	78	112304.11	0.008	78	112304.11	0.008
gesa2	199	25697002.26	0.321	298	25707807.56	0.279	298	25707807.56	0.279
gesa2o	285	25672283.61	0.417	505	25731447.52	0.188	593	25733442.14	0.180
gesa3	96	27917645.34	0.262	201	27926341.84	0.231	243	27930291.92	0.217
gesa3o	120	27925108.93	0.236	175	27950003.17	0.147	184	27950849.64	0.144
khb05250	63	105874167.26	0.997	135	106635740.29	0.285	180	106897242.76	0.040
misc06	0	12841.69	0.071	0	12841.69	0.071	0	12841.69	0.071
mod011	1	-62116832.14	13.854	41	-61920305.56	13.493	104	-61770780.90	13.219
modglob	89	20638589.07	0.491	329	20703526.03	0.178	620	20711278.29	0.141
noswot	41	-43.00	0.000	41	-43.00	0.000	41	-43.00	0.000
pk1	0	0.00	100.000	0	0.00	100.000	0	0.00	100.000
pp08a	174	6689.25	8.990	570	7158.35	2.608	759	7215.02	1.836
pp08aCUTS	197	6690.71	8.970	553	7138.04	2.884	918	7187.45	2.212
qiu	0	-931.64	601.149	0	-931.64	601.149	0	-931.64	601.149
rentacar	18	29036336.63	4.350	96	29113393.96	4.096	88	29094997.33	4.156
rgn	43	64.48	21.559	80	70.80	13.874	80	70.80	13.874
rout	15	981.86	8.881	15	981.86	8.881	15	981.86	8.881
set1ch	493	50957.36	6.565	973	51738.00	5.134	1551	51867.98	4.895
vpm1	211	19.50	2.500	429	19.75	1.250	595	19.80	1.000
vpm2	246	12.47	9.313	449	12.58	8.482	639	12.61	8.299
Average	125		5.806	283		4.793	375		4.675

Table 12: Our MIR results (CRITERION = B, MULTIPLY = TRUE, Preprocessing = True, Row to aggregate: select randomly).

	MAXAGGR = 2			MAXAGGR = 4			MAXAGGR = 6		
Instance	Cuts	LP Soln	Gap	Cuts	LP Soln	Gap	Cuts	LP Soln	Gap
arki001	1	7579599.81	0.016	6	7579599.81	0.016	14	7579599.81	0.016
bell3a	11	869442.13	1.023	15	869660.65	0.998	15	869660.65	0.998
bell5	11	8610266.93	3.972	26	8616747.65	3.900	14	8612370.86	3.948
blend2	145	7.09	6.718	434	7.11	6.488	698	7.10	6.553
danoint	82	62.68	4.556	196	62.68	4.559	187	62.68	4.560
dcmulti	12	184173.98	2.130	20	184173.98	2.130	32	184174.52	2.130
egout	96	393.80	30.680	147	393.87	30.670	210	395.70	30.348
fixnet6	669	3495.17	12.248	1820	3547.80	10.926	2092	3444.71	13.515
flugpl	0	1167185.73	2.856	0	1167185.73	2.856	0	1167185.73	2.856
gen	78	112304.11	0.008	78	112304.11	0.008	78	112304.11	0.008
gesa2	199	25697002.26	0.321	298	25707807.56	0.279	298	25707807.56	0.279
gesa2o	281	25624659.31	0.602	545	25694895.82	0.330	669	25708206.16	0.278
gesa3	96	27917645.34	0.262	195	27926341.84	0.231	220	27929166.93	0.221
gesa3o	135	27925108.93	0.236	215	27951013.91	0.143	219	27951305.97	0.142
khb05250	62	105874167.26	0.997	125	106633189.56	0.287	197	106889255.93	0.048
misc06	0	12841.69	0.071	0	12841.69	0.071	0	12841.69	0.071
mod011	0	-62121982.55	13.863	10	-62081910.74	13.790	17	-62076829.41	13.780
modglob	89	20638589.07	0.491	329	20703526.03	0.178	620	20711278.29	0.141
noswot	41	-43.00	0.000	41	-43.00	0.000	41	-43.00	0.000
pk1	0	0.00	100.000	0	0.00	100.000	0	0.00	100.000
pp08a	158	6744.42	8.239	663	7181.07	2.298	1155	7203.67	1.991
pp08aCUTS	197	6690.71	8.970	553	7138.04	2.884	918	7187.45	2.212
qiu	0	-931.64	601.149	0	-931.64	601.149	0	-931.64	601.149
rentacar	7	29036336.63	4.350	13	29036336.63	4.350	20	29036336.63	4.350
rgn	43	64.48	21.559	80	70.80	13.874	80	70.80	13.874
rout	15	981.86	8.881	15	981.86	8.881	15	981.86	8.881
set1ch	496	50957.36	6.565	980	51738.00	5.134	1562	51867.98	4.895
vpm1	211	19.50	2.500	429	19.75	1.250	595	19.80	1.000
vpm2	246	12.47	9.313	449	12.58	8.482	639	12.61	8.299
Average	128		5.824	294		4.808	406		4.823

Table 13: Our MIR results (CRITERION = B, MULTIPLY = TRUE, Preprocessing = True, Row to aggregate: select first found).

	MAXAGGR = 2			MAXAGGR = 4			MAXAGGR = 6		
Instance	Cuts	LP Soln	Gap	Cuts	LP Soln	Gap	Cuts	LP Soln	Gap
arki001	22	7579698.35	0.015	38	7579733.14	0.014	62	7579715.84	0.014
bell3a	8	868392.08	1.143	29	868728.97	1.104	26	868736.11	1.104
bell5	6	8610068.73	3.974	9	8610068.73	3.974	8	8610068.73	3.974
blend2	0	6.92	8.992	0	6.92	8.992	0	6.92	8.992
danoint	3	62.64	4.618	10	62.64	4.617	23	62.64	4.612
dcmulti	6	184044.83	2.198	68	184495.03	1.959	103	185200.51	1.584
egout	81	287.38	49.414	148	292.16	48.572	271	297.99	47.547
fixnet6	425	2141.92	46.223	1737	2558.61	35.762	2075	2110.93	47.002
flugpl	0	1167185.73	2.856	0	1167185.73	2.856	0	1167185.73	2.856
gen	55	112300.13	0.011	55	112300.13	0.011	55	112300.13	0.011
gesa2	148	25691693.98	0.342	154	25691778.37	0.342	154	25691778.37	0.342
gesa2o	91	25512531.76	1.037	109	25513328.56	1.034	136	25516083.57	1.023
gesa3	56	27917134.94	0.264	69	27914282.80	0.274	73	27914282.80	0.274
gesa3o	40	27861436.55	0.463	55	27865995.79	0.447	54	27866168.14	0.446
khb05250	7	97499588.52	8.828	15	98574156.01	7.823	37	99582158.56	6.881
misc06	0	12841.69	0.071	0	12841.69	0.071	0	12841.69	0.071
mod011	58	-61832496.78	13.332	525	-58955515.34	8.059	708	-58962069.80	8.071
modglob	39	20459313.81	1.356	159	20501850.00	1.151	281	20514549.22	1.089
noswot	41	-43.00	0.000	41	-43.00	0.000	41	-43.00	0.000
pk1	0	0.00	100.000	0	0.00	100.000	0	0.00	100.000
pp08a	154	6660.25	9.384	510	7011.07	4.611	895	7034.28	4.295
pp08aCUTS	172	5577.58	24.115	593	6728.61	8.454	891	6769.11	7.903
qiu	0	-931.64	601.149	0	-931.64	601.149	0	-931.64	601.149
rentacar	22	29036336.63	4.350	120	29165351.32	3.925	106	29091205.23	4.169
rgn	59	67.60	17.764	108	74.54	9.316	108	74.54	9.316
rout	8	981.86	8.881	8	981.86	8.881	8	981.86	8.881
set1ch	392	46233.22	15.227	1010	48000.32	11.987	1601	47177.13	13.496
vpm1	85	17.12	14.378	153	17.49	12.556	214	17.64	11.785
vpm2	116	11.47	16.578	261	11.72	14.764	345	11.74	14.634
Average	79		9.839	229		7.752	317		8.091

Table 14: Our MIR results (CRITERION = C, MULTIPLY = TRUE, Preprocessing = True, Row to aggregate: select randomly).

	MAXAGGR = 2			MAXAGGR = 4			MAXAGGR = 6		
Instance	Cuts	LP Soln	Gap	Cuts	LP Soln	Gap	Cuts	LP Soln	Gap
arki001	10	7579683.73	0.015	12	7579683.73	0.015	37	7579683.73	0.015
bell3a	6	868545.16	1.125	6	868545.16	1.125	6	868545.16	1.125
bell5	6	8610068.73	3.974	11	8610672.98	3.967	9	8612221.06	3.950
blend2	0	6.92	8.992	0	6.92	8.992	0	6.92	8.992
danoint	1	62.64	4.618	21	62.64	4.618	23	62.64	4.615
dcmulti	11	184147.48	2.144	25	184147.48	2.144	48	184147.54	2.144
egout	87	264.08	53.515	215	273.42	51.872	352	278.17	51.035
fixnet6	426	2141.92	46.223	1744	2556.80	35.807	2043	2110.62	47.009
flugpl	0	1167185.73	2.856	0	1167185.73	2.856	0	1167185.73	2.856
gen	55	112300.13	0.011	55	112300.13	0.011	55	112300.13	0.011
gesa2	148	25691693.98	0.342	154	25691778.37	0.342	154	25691778.37	0.342
gesa2o	97	25513312.68	1.034	136	25515065.47	1.027	144	25515511.32	1.025
gesa3	56	27917134.94	0.264	69	27914282.80	0.274	73	27914282.80	0.274
gesa3o	42	27861436.55	0.463	49	27865995.79	0.447	51	27866253.78	0.446
khb05250	7	97499588.52	8.828	23	98481790.31	7.909	48	100117073.30	6.380
misc06	0	12841.69	0.071	0	12841.69	0.071	0	12841.69	0.071
mod011	55	-61871726.91	13.404	332	-59660630.69	9.352	411	-60086600.19	10.132
modglob	39	20459313.81	1.356	159	20501850.00	1.151	281	20514549.22	1.089
noswot	41	-43.00	0.000	41	-43.00	0.000	41	-43.00	0.000
pk1	0	0.00	100.000	0	0.00	100.000	0	0.00	100.000
pp08a	154	6493.38	11.655	824	7082.38	3.641	1428	7113.20	3.222
pp08aCUTS	172	5577.58	24.115	593	6728.61	8.454	891	6769.11	7.903
qiu	0	-931.64	601.149	0	-931.64	601.149	0	-931.64	601.149
rentacar	10	29036336.63	4.350	24	29036336.63	4.350	35	29036336.63	4.350
rgn	59	67.60	17.764	108	74.54	9.316	108	74.54	9.316
rout	8	981.86	8.881	8	981.86	8.881	8	981.86	8.881
set1ch	390	45802.74	16.016	943	45730.51	16.149	1462	45892.55	15.852
vpm1	85	17.12	14.378	153	17.49	12.556	214	17.64	11.785
vpm2	116	11.47	16.578	261	11.72	14.764	345	11.74	14.634
Average	78		10.114	228		8.080	316		8.364

Table 15: Our MIR results (CRITERION = C, MULTIPLY = TRUE, Preprocessing = True, Row to aggregate: select first found).