Tilburg University

Problem Reduction in Set Partitioning Problems

van Krieken, M.G.C.; Fleuren, H.A.; Peeters, M.J.P.

Publication date:
2003

[Link to publication in Tilburg University Research Portal](#)

Citation for published version (APA):

van Krieken, M. G. C., Fleuren, H. A., & Peeters, M. J. P. (2003). *Problem Reduction in Set Partitioning Problems*. (CentER Discussion Paper; Vol. 2003-80). Operations research.

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

Take down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

No. 2003–80

**PROBLEM REDUCTION IN SET
PARTITIONING PROBLEMS**

By M.G.C. van Krieken, H. Fleuren, M.J.P. Peeters

August 2003

ISSN 0924-7815

Problem reduction in set partitioning problems

Maaïke van Krieken*, Hein Fleuren, René Peeters

CentER Applied Research, Tilburg University, PO Box 90153, 5000 LE Tilburg, The Netherlands

* Corresponding author, e-mail: m.g.c.vankrieken@uvt.nl

Abstract

In this paper, several problem reduction techniques are discussed that can be used to reduce the solution time of set partitioning problems. These techniques can be applied in any solution algorithm for set partitioning problems. Besides a short review of the existing literature on preprocessing set partitioning problems, we also present several new techniques. The value of these techniques is illustrated by various computational experiments.

1. Introduction

Given a collection of subsets of a certain ground set and cost associated to these subsets, the set partitioning problem is the problem of finding a minimal cost partition of the ground set. More formally, the set partitioning problem can be written as follows:

$$\begin{aligned} \min \quad & \sum_{j \in J} c_j \cdot x_j \\ \text{Subject to:} \quad & \sum_{j \in J} a_{rj} \cdot x_j = 1 & r \in R \\ & x_j \in \{0,1\} & j \in J \end{aligned}$$

Here, R is the set of *constraints* or *rows* (ground set) and J is the collection of *subsets* or *columns*. The matrix $A = \{a_{rj}\}$ is defined such that a_{rj} is equal to 1 if subset j contains row r and 0 otherwise. The cost of a subset j is given by c_j . Furthermore, we define $R(j)$ as the set of rows that are contained in subset j and $J(r)$ as the collection of subsets that contain row r .

The set partitioning problem is an NP-hard combinatorial optimization problem that has many real-life applications. Problems like vehicle routing, facility location and airline crew scheduling can be formulated as a set partitioning problem. For this reason, the set partitioning problem has been studied extensively in the past decades. See Balas & Padberg

(1976) for a survey of the literature. More recently there has also been much attention for different solution techniques for set partitioning problems. See for example Hoffman & Padberg (1993), where an LP-based algorithm is applied, Beasley & Cao (1996), who use a dynamic programming approach, Chu & Beasley (1995), who discuss a genetic algorithm and Fleuren (1988), who uses a branch and bound approach with lagrangean relaxation for lower bound calculations.

In this paper we study problem reduction techniques that can be used in any of these solution methods. The goal of problem reduction techniques is to reformulate the problem such that the optimal solution to the original problem is the same as the optimal solution to the reformulated problem, where the reformulated problem is easier to solve than the original problem. This results mostly in removing redundant subsets, i.e. subsets that can never be part of an optimal solution. This can, however, also result in removing redundant constraints, i.e. constraints that are automatically satisfied when the other constraints are satisfied. Finally, as will be shown in the following, problem reduction can also be achieved by combining constraints such that, although the number of subsets can increase slightly, the number of constraints decreases. Applying problem reduction techniques can reduce the calculation time of solution algorithms for set partitioning problems, like branch and bound or branch and cut, significantly. However, we have to take into account the obvious trade-off between the time it takes to perform problem reduction techniques and the solution time that is saved. This will be illustrated by the computational experiments described in section 5.

In section 2, several preprocessing techniques will be discussed. In section 3, we will discuss a technique that aims at reducing the number of constraints by allowing a small increase in subsets. Sections 4 and 5 describe several computational results. A final summary is given in section 6.

2. Preprocessing techniques

In this section several different preprocessing rules for the set partitioning problem are discussed. Some of these techniques can be found in the existing literature on set partitioning problems, see for example Müller (1998), Borndörfer (1998) and Hoffman & Padberg (1993). In this section we generalize some of these well-known techniques and also present the new CUT preprocessing technique. In section 4 computational results considering these preprocessing techniques will be discussed.

Equal k-subsets

If a subset j can be represented by a combination of k other subsets, $k > 0$, with less cost, then subset j can be removed from the problem. More formally:

If $R(j) = \bigcup_{i \in K} R(i)$, $R(i_1) \cap R(i_2) = \emptyset \ \forall \{i_1, i_2\} \in K$ and $c(j) \geq \sum_{i \in K} c(i)$ for $j \in J, K \subset J \setminus j$ then

subset j can be removed from the problem.

The well-known “equal columns” preprocessing rule (see for example Hoffman & Padberg, 1993) is part of this set of preprocessing techniques, with $k = 1$. When the number k increases, finding equal k -subsets becomes more complex. This is caused by the fact that we have to check more combinations of subsets, whereas the collection of subsets usually is quite large. In section 4, we will discuss computational results of this rule for $k = 1$ and $k = 2$.

k-Rowsets

If there is a set of k rows, $r_1, \dots, r_k$, $k > 1$, for which holds that there is no subset j for which $r_1 \in R(j)$ and $r_2, \dots, r_k \notin R(j)$, than all subsets c for which $r_1 \notin R(c)$ and $r_2, \dots, r_k \in R(c)$ can be deleted. More formally:

Let $r_1, \dots, r_k \in R$. If $\nexists j \in J : r_1 \in R(j), r_2, \dots, r_k \notin R(j)$ than we can delete all subsets in $\{c \in J \mid r_1 \notin R(c), r_2, \dots, r_k \in R(c)\}$.

The well-known “contained rows” preprocessing rule (see for example Hoffman & Padberg, 1993) is a variant of the k -rowsets rule with $k = 2$. This is the case when one row r is contained in another row s . In this case, the k -rowset rule says that we can delete all subsets that contain s , but not r . However, additional advantage in the case $k = 2$ is that we are left with two equal rows in the constraint matrix. This means that one of the two constraints in question has become redundant and can be deleted as well. The k -rowset technique does not incorporate this observation. This leads to a new preprocessing technique, that checks for equal rows and will be treated next. In section 4 we discuss computational results for the k -rowsets rule for $k = 2$ and $k = 3$. Again, the problem of finding k -rowsets becomes more difficult when k increases, since more combinations have to be checked.

Equal rows

If two constraints are identical, one of them can be deleted from the problem. More formally:
If $J(r) = J(s)$ than row s can be deleted.

The equal rows technique in itself will not have much effect on a real-life set partitioning problem, since equal constraints are not likely to appear. Some of the other

techniques described in this section, however, like the k-rowset and clique techniques, can result in equal rows.

Clique rule

If all subsets that contain row r have one or more elements in common with a subset j that does not contain row r , then we can remove subset j , since choosing this subset in a solution set will leave constraint r unsatisfiable. Another way to see this (Hoffman & Padberg, 1993) is as follows. Derive a graph from the set partitioning problem where the nodes of the graph correspond to the subsets and two nodes are connected if they share at least one element. A trivial clique C_r in such a graph is the set of all nodes (subsets) containing a certain element r of the ground set R . This implies that every feasible solution contains only one element of this clique. If we can find a clique C that properly subsumes C_r , then every subset in $C \setminus C_r$ can be removed.

Relation clique – k-rowset

For every value of k , the k-rowset preprocessing technique is a special case of the clique preprocessing technique, meaning that any reductions found by the former will also be found by the latter. Suppose that for $r_1, \dots, r_k$, $k > 1$ there is no subset j for which $r_1 \in R(j)$ and $r_2, \dots, r_k \notin R(j)$. This means that every subset that covers row r_1 has at least one element in common with every subset in $\{c \in J \mid r_1 \notin R(c), r_2, \dots, r_k \in R(c)\}$. Therefore, according to the clique preprocessing technique, all subsets in $\{c \in J \mid r_1 \notin R(c), r_2, \dots, r_k \in R(c)\}$ can be deleted. This observation does not necessarily imply that the k-rowset rule has become obsolete. Since the clique preprocessing techniques covers a more general situation than the k-rowset rule, performing this rule also takes more time. In section 5, we will illustrate this with a computational experiment.

CUT rule

For a given set of three rows $\{r, s, t\}$, we define $CS(r, s, t)$ as the set of columns that cover at least two of rows r , s and t :

$$CS(r, s, t) = \{j \in J \mid |R(j) \cap \{r, s, t\}| \geq 2\}$$

The CUT rule says that if we can find a row w for which $J(w) \subseteq CS(r, s, t)$ then we can delete all columns in the set $CS(r, s, t) - J(w)$.

More intuitively, we can explain this as follows. For the set of rows (r,s,t) , we can discern four types of subsets:

- Subsets that cover none of these rows: $T_0(r,s,t) = \{j \in J \mid |R(j) \cap \{r,s,t\}| = 0\}$
- Subsets that cover one of these rows: $T_1(r,s,t) = \{j \in J \mid |R(j) \cap \{r,s,t\}| = 1\}$
- Subsets that cover two of these rows: $T_2(r,s,t) = \{j \in J \mid |R(j) \cap \{r,s,t\}| = 2\}$
- Subsets that cover all three rows: $T_3(r,s,t) = \{j \in J \mid |R(j) \cap \{r,s,t\}| = 3\}$

From all the subsets that are incorporated in $T_2(r,s,t)$ and $T_3(r,s,t)$ we cannot take more than one subset in a solution. This actually forms a cut to the linear programming formulation of the problem, as is illustrated by the following example:

$$\begin{array}{l} r \left[\begin{array}{cccccccc} 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{array} \right] = 1 \\ s \left[\begin{array}{cccccccc} 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \end{array} \right] = 1 \\ t \left[\begin{array}{cccccccc} 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{array} \right] = 1 \\ \left[\begin{array}{cccccccc} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{array} \right] \leq 1 \end{array}$$

More formally, this cut is formed by first multiplying rows r , s and t by $\frac{1}{2}$ and then adding them. In the example, this would give:

$$\left[\begin{array}{cccccccc} 1\frac{1}{2} & 1 & 1 & 1 & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 0 \end{array} \right] = 1\frac{1}{2}$$

Since all variables are positive, this leads to:

$$\left[\begin{array}{cccccccc} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{array} \right] \leq 1\frac{1}{2}$$

And, finally, since all variables are integer:

$$\left[\begin{array}{cccccccc} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{array} \right] \leq 1$$

Generally, this cut covers all subsets $j \in T_2(r,s,t) \cup T_3(r,s,t)$. If we can find a row w that is contained in this cut, we can delete all subsets that are in the cut, but not in $J(w)$.

3. Row combinations

In this section we will discuss a new technique that is designed to reduce the number of constraints in the problem. To this end we allow a small increase in the number of subsets. The increase in subsets that we allow is given by a percentage of the total number of subsets at the start of the technique. Below, we will first describe the technique. In the next section we will discuss some results of this technique.

When we say that we combine two rows r_1 and r_2 we mean the following:

For every subset $j_1 \in J(r_1)$, $j_1 \notin J(r_2)$

For every subset $j_2 \in J(r_2)$, $j_2 \notin J(r_1)$

If $R(j_1) \cap R(j_2) = \emptyset$

Make a new subset j_3 for which $R(j_3) = R(j_1) \cup R(j_2)$ and $c_{j_3} = c_{j_1} + c_{j_2}$

When all possible combinations of subsets are made, all subsets that belong either to $J(r_1)$ or to $J(r_2)$, but not to both, can be removed from the problem. Also, one of the two rows can be removed from the problem. This makes the technique particularly interesting for pairs of rows that differ only on a few elements, since in that case we only add a small amount of subsets, while we can remove one row. It can even be the case that the number of rows as well as the number of subsets of the problem decreases. Note that the contained rows preprocessing rule actually is a special case of this technique and that the reductions that can be achieved by this preprocessing rule, will also be achieved by performing the row combination technique.

When rows are combined, it is important to administrate these combinations in such a way that when a solution to the problem is found, we can reconstruct the original subsets of which this solution consists.

The technique works as follows:

Step 0: $\text{Max_growth} = \frac{P}{100} \cdot \text{number of subsets}$.

Step 1: For each $r_1, r_2 \in R$ we define:

$$C(r_1, r_2) = \{j \in J(r_1) \mid j \notin J(r_2)\} \text{ and}$$

$$f(r_1, r_2) = |C(r_1, r_2)| \cdot |C(r_2, r_1)| - |C(r_1, r_2)| - |C(r_2, r_1)|$$

This function gives an upperbound on the increase in number of subsets when rows r_1 and r_2 are combined. Now let $\{s, t\}$ be the set of rows for which $f(r_1, r_2)$ is minimal.

If $(f(s, t) > \text{Max_growth})$ then stop.

Step 2: Combine rows s and t . Now delete all subsets $k \in \{j \in J(s) \mid j \notin J(t)\}$ and all subsets $m \in \{j \in J(t) \mid j \notin J(s)\}$. Go to step 1.

This technique uses the parameter p ; a percentage that denotes the maximal allowed growth in the number of subsets. Extensive testing should be used to determine the value of this parameter carefully. In our application of the technique, we allow the number of subsets to grow with a maximum of 0.5 % of the total number of subsets. Note that this technique can also be used as a preprocessing technique to reduce the number of subsets as well as the

number of rows. To achieve this, the parameter p should be given the value 0, meaning that we do not allow the number of subsets to grow. However, the value of $f(s,t)$ is an upper bound on the increase in the number subsets when rows s and t are combined. In general, the actual increase will be smaller. Furthermore, as will be shown by the computational results in the next section, the number of constraints can be reduced significantly if we allow a small increase in the number of subsets. Besides, experience shows that the calculation time of a solution algorithm for typical practical set partitioning problems is influenced more by the number of constraints than by the number of subsets. These observations illustrate that it can be useful to take a small but positive value for the parameter p .

4. Individual computational results

In this section, we will discuss computational results of the problem reduction techniques described in this paper. The test set consists of 55 real-life set partitioning problems that stem from the OR-library of Beasley (Beasley, 1990). This is the same set as is used in Hoffman & Padberg (1993). Table 1 shows the problem characteristics of these instances, where the density of a problem denotes the percentage of nonzero's in the constraint matrix. We remark that in Tables 2 to 7, 'CS' stands for constraint reduction, 'SS' denotes subset reduction, and '%CS' and '%SS' denote the percentage constraint and subset reductions, respectively.

Equal k- subsets

Table 2 shows the results of applying the equal k -subsets preprocessing technique on the test set for $k = 1$ and $k = 2$. Since many real-life instances of set partitioning problems are generated by column generation, double columns occur frequently, which accounts for the large reductions in the case $k = 1$. When $k = 2$, we see that the preprocessing technique takes much more time, but also that the reductions are much larger. However, the calculation times in this case are too large for this preprocessing technique to be useful in a solution algorithm.

k-Rowsets and equal rows

Table 3 shows the results of applying the k -rowset rule and the equal rows rule consecutively for $k = 2$ and $k = 3$. We first consider the first 2 columns, which give the results of applying the technique only once for every set of k rows. As can be seen, although the rule is not effective on all tested instances, the reductions found are substantial, while the calculation time is relatively small for all tested instances. Note that the implementation of the 3-rowset

technique also includes the search for 2-rowsets, such that the reductions found for $k = 3$ are larger than those found for $k = 2$. When k is increased from 2 to 3 we see that there is an increase in the reductions found for a few instances, but also that the average calculation time increases. Note that for some instances, the time needed to perform the 3-rowset technique is smaller than the time needed for the 2-rowset technique, which is due to a difference in implementation. However, on average, the 2-rowset technique is performed considerably faster.

Note that the order in which rows are considered can influence the results of these preprocessing techniques. We illustrate this with a small example. Suppose we have four rows: r_1, r_2, r_3 and r_4 . Furthermore, we have columns $x_1, \dots, x_n, y_1, \dots, y_m$ and v_1, v_2 such that:

$$J(r_1) = \{v_1, y_1, \dots, y_m\}$$

$$J(r_2) = \{v_2, y_1, \dots, y_m\}$$

$$J(r_3) = \{v_1, x_1, \dots, x_n\}$$

$$J(r_4) = \{x_1, \dots, x_n\}$$

Now apply the 2-rowset technique on this set of rows. If we first consider rows r_1 and r_2 , we conclude that the rule does not apply on these rows. However, if we consider rows r_3 and r_4 , we conclude that subset v_1 can be deleted, since when we choose this subset in a solution set, constraint r_4 cannot be satisfied. If we again consider rows r_1 and r_2 after deletion of column v_1 , we find that we can delete subset v_2 as well. Note that after these subset reductions, we can also delete rows r_2 and r_4 . This example indicates that more reductions can be gained by applying the k -rowset technique iteratively. Note that this also holds for $k > 2$. For $k = 2$, this observation is illustrated by the last column in Table 3, where the results are given of applying the 2-rowset technique iteratively on the entire row set, until no more reductions are achieved. As can be seen, the reductions found are larger in some cases, while the calculation time is only moderately larger.

Clique and equal rows

In Table 4, the results are given of applying the clique rule, followed by the equal rows rule. Again, the sequence in which rows are considered can influence the reductions found by the clique preprocessing technique. Therefore, Table 4 shows the results of applying the clique and equal rows techniques once as well as the results of applying them iteratively until no more reductions are found. The clique technique reduces the set partitioning tableau significantly for most instances, but also uses a considerable amount of calculation time.

Applying the technique iteratively leads to more reductions in some cases, but the increase in calculation time is relatively large compared to the size of the extra reductions.

CUT rule

In Table 5, the results of the CUT preprocessing rule are given. Although this rule does not achieve problem reduction for all instances, the calculation time is very small for all instances. Moreover, for the problems where reductions are found, these reductions are relatively large.

Row combinations

Table 6 shows the results of the row combination technique, for different values of the parameter p . As can be seen, the technique acts as a pure reduction technique when the parameter has value 0. However, when the parameter is given a small but positive value, the row reductions found are considerable, while most instances also show a reduction in subsets. Only for a very small number of instances the number of subsets increases. Another observation is that the calculation time of the technique increases with an increase in the value of the parameter. This is caused by the additional effort needed to add subsets to the tableau and administrate the origin of these subsets.

5. Combined computational results

This section will be divided into two parts. In the first part we will examine the relation between the 2-rowset technique on the one side and the clique and row combination techniques on the other side. The second part will discuss the total reductions achieved when we apply a set of preprocessing techniques on the test set and how these reductions influence the calculation time of the Cplex solver on these problems.

Relation 2-rowset, clique and row combinations

As we mentioned before, the reductions we find with the 2-rowset preprocessing techniques can also be found by applying the clique- or row combinations technique. However, since the 2-rowset rule is less general, it can be performed much faster and therefore can still have use within a set partitioning algorithm. We will illustrate this with two computational experiments. In Table 7, results are given of applying 2-rowset, equal rows, clique and equal rows successively on our test set. When we compare these results to the results of applying

just the combination clique and equal rows, we see that the reductions found are equally large or larger, while the average calculation time is smaller. This is caused by the fact that the calculation time of the 2-rowset technique is very small compared to the time the clique technique needs, while considerable reductions are achieved. Due to these reductions, the calculation time of the clique technique is much smaller than when we applied it on the original problems. Table 7 also shows the results of applying 2-rowset, equal rows and the row combination technique successively on our test set. Here we took $p = 0.00$. Again, we see that the reductions found are the same as in the case where only the row combination technique is applied, but that the average calculation time is much lower.

The trade-off between solution time and preprocessing time

In this section we will examine the effect that the reductions, found by preprocessing techniques, can have on the solution time of set partitioning problems. To this end, we compare the solution time of the well-known commercial solver CPLEX on the original problems with the time needed to solve the preprocessed problems. The calculations are made with the Cplex 8.0 solver, used within the AIMMS modeling environment. Due to memory issues, we had to leave one of the problems out of this comparison. In order to make the comparison pure, we turned off the preprocessing option incorporated in Cplex. We will discuss 6 different sequences of preprocessing techniques:

1. Equal 1-subsets, 2-rowsets, equal rows
2. Equal 1-subsets, 2-rowsets, equal rows, row combinations ($p = 0.5$)
3. Equal 1-subsets, 2-rowsets, equal rows, clique, equal rows
4. Equal 1-subsets, 2-rowsets, equal rows, CUT
5. Equal 1-subsets, 2-rowsets, equal rows, clique, equal rows, CUT
6. Equal 1-subsets, 2-rowsets, equal rows, clique, equal rows, CUT, row combinations ($p = 0.5$)

Table 8 shows a summary of the results of these 6 sequences. As can be seen, in all situations the number of subsets are reduced for all instances. Also, the number of instances with constraint reduction is much higher in the cases where the row combination technique is used. Furthermore, we see that the percentage decrease in solution time varies from 4% to 27% and is positive in all cases, meaning that, in all 6 cases, the total time of solving the original instances is larger than the total preprocessing time plus the time of solving the preprocessed problems. The results illustrate an obvious trade-off between the preprocessing techniques. For example, when we compare sequences 1 and 4, we see that the application of the CUT preprocessing rule decreases the total percentage benefit of the preprocessing sequences on

the solution time of Cplex. However, when we compare sequences 3 and 5, we see that the total percentage benefit increases due the CUT preprocessing technique, when it is preceded by the clique preprocessing rule.

Note that in a solution algorithm, the preprocessing techniques can be used throughout the algorithm, and do not have to be concentrated at the start of the algorithm. Furthermore, application of a technique can be repeated. This indicates that the total benefit of the preprocessing techniques can be higher within a solution algorithm.

6. Conclusions

In this paper, we discussed several problem reduction techniques for set partitioning problems, some of which have not been treated in literature before. Several computational experiments have been described to illustrate the use of these techniques. Six experiments are performed to examine the joint benefit of the preprocessing techniques. The results of these experiments are summarized in Table 8. As can be seen, the average subset reduction is around 25% for all six sequences, while the average constraint reduction varies from 5% to 14%. Generally, the time of performing the preprocessing techniques grows with the number of subsets in the problem. However, in our experience, the time needed to perform preprocessing is generously compensated by the reductions that are accomplished. This is illustrated by the computational results. Consider the experiment where we applied the sequence: equal 1-subsets, 2-rowsets, equal rows, row combination technique (sequence 2). The total time of Cplex on the 54 instances is equal to 221 seconds, while the total preprocessing time is only 29 seconds and the time of solving the preprocessed problems with Cplex is equal to 132 seconds. This means that the total solution time is reduced by 60 seconds, or 27%. Note that the reduction in solution time achieved by preprocessing can be much higher when applied in other solvers. For example, the set partitioning solver that is developed by the authors, which applies a combination of lagrangean relaxation, primal and dual heuristics and branch and bound, saves 95% on the total solution time of the test set by applying the problem reduction techniques described in this paper.

References

- Balas, E. and M.W. Padberg. 1976. Set partitioning: A survey. *SIAM Review*. 18, 710-760.
- Beasley, J.E. 1990. OR-library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*. Vol. 41, no. 11, 1069-1072.
- Beasley, J.E. and B. Cao. 1998. A dynamic programming based algorithm for the crew scheduling problem. *Computers Operations Research*. 25, 567-582.
- Borndörfer, R. 1998. Aspects of set packing, partitioning and covering. Dissertation, Technical University of Berlin, Germany.
- Chu, P.C. and J.E. Beasley. 1995. A genetic algorithm for the set partitioning problem. Technical report, Imperial College London.
- Fleuren, H.A. 1988. A computational study of the set partitioning approach for vehicle routing and scheduling problems. *Dissertation, University of Twente, The Netherlands*.
- Hoffman, K.L. and M. Padberg. 1993. Solving airline crew scheduling problems by branch-and-cut. *Management Science*. 39, 657-682.
- Müller, T. 1998. Solving set partitioning problems with constraint programming. *Proceedings of the Sixth International Conference on the Practical Application of Prolog and the Fourth International Conference on the Practical Application of Constraint Technology (PAPPACT)*. 313-332.

Table 1: The problem characteristics of the 55 instances

Problem	#Subsets	#Constraints	#Elements	Density
nw41	197	17	740	22.10%
nw32	294	19	1357	24.29%
nw40	404	19	2069	26.95%
nw08	434	24	2332	22.39%
nw15	467	31	2830	19.55%
nw21	577	25	3591	24.89%
nw22	619	23	3399	23.87%
nw12	626	27	3380	20.00%
nw39	677	25	4494	26.55%
nw20	685	22	3722	24.70%
nw23	711	19	3350	24.80%
nw37	770	19	3778	25.82%
nw26	771	23	4215	23.77%
nw10	853	24	4336	21.18%
nw34	899	20	5045	28.06%
nw43	1072	18	4859	25.18%
nw42	1079	23	6533	26.32%
nw28	1210	18	8553	39.27%
nw25	1217	20	7341	30.16%
nw38	1220	23	9071	32.33%
nw27	1355	22	9395	31.52%
nw24	1366	19	8617	33.20%
nw35	1709	23	10494	26.70%
nw36	1783	20	13160	36.90%
nw29	2540	18	14193	31.04%
nw30	2653	26	20436	29.63%
nw31	2662	26	19977	28.86%
nw19	2879	40	25193	21.88%
nw33	3068	23	21704	30.76%
nw09	3103	40	20111	16.20%
nw07	5172	36	41187	22.12%
aa02	5198	531	36359	1.32%
nw06	6774	50	61555	18.17%
aa04	7195	426	52121	1.70%
aa06	7292	646	51728	1.10%
kl01	7479	55	56242	13.67%
aa05	8308	801	65953	0.99%
aa03	8627	825	70806	0.99%
nw11	8820	39	57250	16.64%
aa01	8904	823	72965	1.00%
nw18	10757	124	91028	6.82%
us02	13635	100	192716	14.13%
nw13	16043	51	104541	12.78%
us04	28016	163	297538	6.52%
kl02	36699	71	212536	8.16%
nw03	43749	59	363939	14.10%
nw01	51975	135	410894	5.86%
us03	85552	77	1211929	18.40%
nw04	87482	36	636666	20.22%
nw02	87879	145	721736	5.66%
nw17	118607	61	1010039	13.96%
nw14	123409	73	904910	10.04%
nw16	148633	139	1501820	7.27%
nw05	288507	71	2063641	10.07%
us01	1053137	145	13636541	8.93%

Table 2: The results of the equal k -subsets preprocessing rule for $k = 1$ and $k = 2$

Problem	Equal 1 - subsets			Equal 2 - subsets		
	SS	% SS	Time (s)	SS	% SS	Time (s)
nw41	20	10%	0.000	109	55%	0.000
nw32	42	14%	0.000	150	51%	0.015
nw40	68	17%	0.000	176	44%	0.000
nw08	78	18%	0.000	243	56%	0.016
nw15	2	0%	0.000	8	2%	0.000
nw21	151	26%	0.000	333	58%	0.000
nw22	88	14%	0.000	262	42%	0.016
nw12	172	27%	0.000	303	48%	0.000
nw39	110	16%	0.000	347	51%	0.000
nw20	119	17%	0.000	299	44%	0.031
nw23	237	33%	0.000	428	60%	0.000
nw37	131	17%	0.000	421	55%	0.015
nw26	229	30%	0.000	412	53%	0.016
nw10	194	23%	0.000	504	59%	0.016
nw34	149	17%	0.000	469	52%	0.032
nw43	89	8%	0.000	528	49%	0.016
nw42	184	17%	0.000	397	37%	0.031
nw28	385	32%	0.000	600	50%	0.016
nw25	373	31%	0.000	861	71%	0.015
nw38	309	25%	0.000	444	36%	0.031
nw27	429	32%	0.000	805	59%	0.015
nw24	440	32%	0.000	946	69%	0.016
nw35	306	18%	0.000	958	56%	0.032
nw36	375	21%	0.000	484	27%	0.031
nw29	506	20%	0.000	1005	40%	0.063
nw30	769	29%	0.000	1756	66%	0.047
nw31	839	32%	0.000	1628	61%	0.047
nw19	734	25%	0.000	1545	54%	0.047
nw33	653	21%	0.000	1524	50%	0.093
nw09	798	26%	0.015	1972	64%	0.063
nw07	2064	40%	0.016	3567	69%	0.141
aa02	0	0%	0.000	160	3%	0.219
nw06	797	12%	0.000	2095	31%	0.391
aa04	0	0%	0.000	197	3%	0.406
aa06	0	0%	0.000	384	5%	0.406
kl01	676	9%	0.015	676	9%	0.515
aa05	0	0%	0.000	264	3%	0.547
aa03	0	0%	0.000	342	4%	0.609
nw11	2332	26%	0.000	5775	65%	0.422
aa01	0	0%	0.000	196	2%	0.640
nw18	2297	21%	0.016	4136	38%	0.954
us02	121	1%	0.047	537	4%	1.531
nw13	5138	32%	0.031	9910	62%	1.328
us04	1026	4%	0.063	9429	34%	5.968
kl02	19965	54%	0.094	19965	54%	6.875
nw03	4785	11%	0.125	14708	34%	21.656
nw01	1906	4%	0.171	16711	32%	29.438
us03	1432	2%	0.407	46078	54%	70.453
nw04	41292	47%	0.312	41313	47%	54.875
nw02	2621	3%	0.312	28161	32%	87.453
nw17	40421	34%	0.531	78070	66%	106.781
nw14	28231	23%	0.515	68694	56%	132.531
nw16	9682	7%	0.656	9682	7%	203.125
nw05	85904	30%	1.469	193578	67%	418.063
us01	408759	39%	5.531	671816	64%	7680.000

Table 3: The results of k -rowset + equal rows for $k = 2$ and $k = 3$

Problem	One time 2-rowset + equal rows					One time 3-rowset + equal rows					2-rowset + equal rows while reduc>0				
	CS	SS	%CS	%SS	Time (s)	CS	SS	%CS	%SS	Time (s)	CS	SS	%CS	%SS	Time (s)
nw41	0	0	0%	0%	0.000	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw32	0	0	0%	0%	0.000	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw40	0	0	0%	0%	0.000	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw08	0	0	0%	0%	0.000	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw15	0	0	0%	0%	0.000	0	58	0%	12%	0.000	0	0	0%	0%	0.000
nw21	0	0	0%	0%	0.000	0	0	0%	0%	0.000	0	0	0%	0%	0.016
nw22	0	0	0%	0%	0.000	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw12	0	0	0%	0%	0.000	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw39	0	0	0%	0%	0.000	0	0	0%	0%	0.000	0	0	0%	0%	0.016
nw20	0	0	0%	0%	0.000	0	28	0%	4%	0.000	0	0	0%	0%	0.000
nw23	0	0	0%	0%	0.000	0	53	0%	7%	0.000	0	0	0%	0%	0.000
nw37	0	0	0%	0%	0.000	0	0	0%	0%	0.015	0	0	0%	0%	0.000
nw26	0	0	0%	0%	0.000	2	122	9%	16%	0.000	0	0	0%	0%	0.000
nw10	0	0	0%	0%	0.000	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw34	0	0	0%	0%	0.000	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw43	0	0	0%	0%	0.000	0	0	0%	0%	0.015	0	0	0%	0%	0.000
nw42	0	0	0%	0%	0.000	0	51	0%	5%	0.000	0	0	0%	0%	0.000
nw28	0	0	0%	0%	0.000	0	315	0%	26%	0.000	0	0	0%	0%	0.016
nw25	0	0	0%	0%	0.000	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw38	0	0	0%	0%	0.000	0	119	0%	10%	0.015	0	0	0%	0%	0.000
nw27	0	0	0%	0%	0.000	0	91	0%	7%	0.000	0	0	0%	0%	0.000
nw24	0	0	0%	0%	0.000	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw35	0	0	0%	0%	0.000	0	126	0%	7%	0.000	0	0	0%	0%	0.000
nw36	0	0	0%	0%	0.000	0	214	0%	12%	0.000	0	0	0%	0%	0.000
nw29	0	0	0%	0%	0.000	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw30	0	0	0%	0%	0.016	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw31	0	0	0%	0%	0.000	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw19	0	0	0%	0%	0.000	0	0	0%	0%	0.016	0	0	0%	0%	0.000
nw33	0	0	0%	0%	0.000	0	62	0%	2%	0.000	0	0	0%	0%	0.000
nw09	0	0	0%	0%	0.016	0	0	0%	0%	0.016	0	0	0%	0%	0.000
nw07	0	0	0%	0%	0.000	0	0	0%	0%	0.000	0	0	0%	0%	0.000
aa02	137	959	26%	18%	0.094	142	1069	27%	21%	0.032	165	1242	31%	24%	0.219
nw06	0	0	0%	0%	0.000	0	0	0%	0%	0.000	0	0	0%	0%	0.000
aa04	77	752	18%	10%	0.094	77	820	18%	11%	0.047	83	995	19%	14%	0.172
aa06	129	1082	20%	15%	0.172	132	1250	20%	17%	0.031	135	1139	21%	16%	0.297
kl01	8	863	15%	12%	0.016	8	905	15%	12%	0.016	8	863	15%	12%	0.015
aa05	225	1516	28%	18%	0.297	234	1699	29%	20%	0.047	256	1877	32%	23%	0.641
aa03	242	1400	29%	16%	0.328	249	1684	30%	20%	0.062	259	1577	31%	18%	0.485
nw11	0	0	0%	0%	0.000	0	0	0%	0%	0.000	0	0	0%	0%	0.000
aa01	188	1152	23%	13%	0.313	190	1252	23%	14%	0.062	203	1217	25%	14%	0.625
nw18	0	0	0%	0%	0.000	0	1	0%	0%	0.016	0	0	0%	0%	0.000
us02	55	2844	55%	21%	0.140	55	5726	55%	42%	0.125	55	2844	55%	21%	0.141
nw13	0	0	0%	0%	0.000	0	0	0%	0%	0.016	0	0	0%	0%	0.000
us04	50	14537	31%	52%	0.266	50	15951	31%	57%	0.156	62	18576	38%	66%	0.390
kl02	2	0	3%	0%	0.047	2	0	3%	0%	0.047	2	0	3%	0%	0.047
nw03	0	0	0%	0%	0.000	0	0	0%	0%	0.062	0	0	0%	0%	0.016
nw01	0	0	0%	0%	0.000	0	168	0%	0%	0.172	0	0	0%	0%	0.000
us03	25	30368	32%	35%	0.891	25	32117	32%	38%	0.859	27	36780	35%	43%	1.125
nw04	0	0	0%	0%	0.000	0	0	0%	0%	0.140	0	0	0%	0%	0.000
nw02	0	0	0%	0%	0.000	0	2	0%	0%	0.328	0	0	0%	0%	0.016
nw17	0	0	0%	0%	0.016	0	0	0%	0%	0.219	0	0	0%	0%	0.016
nw14	0	0	0%	0%	0.016	0	0	0%	0%	0.250	0	0	0%	0%	0.016
nw16	0	0	0%	0%	0.297	0	0	0%	0%	0.297	0	0	0%	0%	0.312
nw05	0	0	0%	0%	0.047	0	0	0%	0%	0.610	0	0	0%	0%	0.031
us01	59	53729	41%	5%	9.219	59	70244	41%	7%	16.797	59	53729	41%	5%	9.296

Table 4: Results of the clique and equal rows preprocessing rules

Problem	One time CL&ER					CL&ER while total reduc > 0				
	CS	SS	%CS	%SS	Time (s)	CS	SS	%CS	%SS	Time (s)
nw41	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw32	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw40	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw08	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw15	0	58	0%	12%	0.000	0	58	0%	12%	0.000
nw21	0	5	0%	1%	0.000	0	5	0%	1%	0.000
nw22	0	12	0%	2%	0.000	0	12	0%	2%	0.000
nw12	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw39	0	2	0%	0%	0.000	0	2	0%	0%	0.000
nw20	0	30	0%	4%	0.000	0	30	0%	4%	0.000
nw23	0	93	0%	13%	0.000	0	93	0%	13%	0.000
nw37	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw26	2	123	9%	16%	0.000	2	126	9%	16%	0.000
nw10	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw34	0	39	0%	4%	0.000	0	39	0%	4%	0.000
nw43	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw42	0	118	0%	11%	0.016	0	118	0%	11%	0.015
nw28	0	384	0%	32%	0.000	0	384	0%	32%	0.016
nw25	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw38	0	232	0%	19%	0.000	0	232	0%	19%	0.015
nw27	0	137	0%	10%	0.000	0	137	0%	10%	0.016
nw24	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw35	0	256	0%	15%	0.000	0	256	0%	15%	0.016
nw36	0	265	0%	15%	0.000	0	265	0%	15%	0.000
nw29	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw30	0	10	0%	0%	0.016	0	10	0%	0%	0.015
nw31	0	125	0%	5%	0.000	0	125	0%	5%	0.015
nw19	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw33	0	112	0%	4%	0.016	0	112	0%	4%	0.031
nw09	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw07	0	0	0%	0%	0.015	0	0	0%	0%	0.015
aa02	136	1169	26%	22%	0.203	169	1350	32%	26%	0.594
nw06	0	0	0%	0%	0.015	0	0	0%	0%	0.031
aa04	73	930	17%	13%	0.234	83	1072	19%	15%	0.797
aa06	119	1145	18%	16%	0.390	142	1390	22%	19%	1.594
kl01	8	905	15%	12%	0.078	8	905	15%	12%	0.125
aa05	216	1697	27%	20%	0.609	268	2062	33%	25%	2.110
aa03	235	1696	28%	20%	0.593	272	1920	33%	22%	2.110
nw11	0	0	0%	0%	0.016	0	0	0%	0%	0.016
aa01	182	1231	22%	14%	0.594	206	1360	25%	15%	1.953
nw18	0	2	0%	0%	0.078	0	2	0%	0%	0.172
us02	55	6194	55%	45%	0.438	55	6719	55%	49%	0.672
nw13	0	0	0%	0%	0.063	0	0	0%	0%	0.062
us04	46	16250	28%	58%	0.891	62	19784	38%	71%	1.516
kl02	2	0	3%	0%	0.421	2	0	3%	0%	0.766
nw03	0	0	0%	0%	0.219	0	0	0%	0%	0.235
nw01	0	168	0%	0%	1.016	0	168	0%	0%	2.141
us03	23	38458	30%	45%	12.641	27	41913	35%	49%	15.797
nw04	0	0	0%	0%	0.484	0	0	0%	0%	0.484
nw02	0	2	0%	0%	2.360	0	2	0%	0%	4.844
nw17	0	0	0%	0%	0.828	0	0	0%	0%	0.828
nw14	0	0	0%	0%	0.906	0	0	0%	0%	0.906
nw16	0	0	0%	0%	3.343	0	0	0%	0%	3.343
nw05	0	0	0%	0%	2.985	0	0	0%	0%	2.969
us01	59	86197	41%	8%	207.703	59	86201	41%	8%	327.985

Table 5: Results of the CUT preprocessing rule

Problem	SS	%SS	Time (s)
nw41	0	0%	0.000
nw32	0	0%	0.000
nw40	0	0%	0.000
nw08	0	0%	0.000
nw15	0	0%	0.015
nw21	0	0%	0.016
nw22	0	0%	0.000
nw12	0	0%	0.016
nw39	0	0%	0.016
nw20	0	0%	0.016
nw23	0	0%	0.015
nw37	0	0%	0.000
nw26	42	5%	0.000
nw10	0	0%	0.000
nw34	0	0%	0.000
nw43	0	0%	0.000
nw42	0	0%	0.000
nw28	0	0%	0.000
nw25	0	0%	0.016
nw38	0	0%	0.000
nw27	0	0%	0.000
nw24	0	0%	0.000
nw35	0	0%	0.000
nw36	0	0%	0.000
nw29	0	0%	0.000
nw30	0	0%	0.000
nw31	0	0%	0.000
nw19	0	0%	0.000
nw33	0	0%	0.000
nw09	0	0%	0.000
nw07	0	0%	0.000
aa02	346	7%	0.297
nw06	0	0%	0.000
aa04	348	5%	0.125
aa06	541	7%	0.281
kl01	235	3%	0.016
aa05	471	6%	0.547
aa03	370	4%	1.031
nw11	0	0%	0.000
aa01	260	3%	0.750
nw18	0	0%	0.000
us02	0	0%	0.656
nw13	0	0%	0.000
us04	11809	42%	1.141
kl02	0	0%	0.016
nw03	0	0%	0.000
nw01	0	0%	0.000
us03	8061	9%	0.344
nw04	0	0%	0.016
nw02	0	0%	0.000
nw17	0	0%	0.000
nw14	0	0%	0.031
nw16	0	0%	0.000
nw05	0	0%	0.062
us01	0	0%	0.109

Table 6: Results of the row combination technique for $p = 0.0$, $p = 0.5$ and $p = 1.0$

Problem	p = 0.0					p = 0.5					p = 1.0				
	CS	SS	%CS	%SS	Time (s)	CS	SS	%CS	%SS	Time (s)	CS	SS	%CS	%SS	Time (s)
nw41	0	0	0%	0%	0.000	2	1	12%	1%	0.000	2	1	12%	1%	0.000
nw32	0	0	0%	0%	0.000	3	2	16%	1%	0.000	3	2	16%	1%	0.000
nw40	0	0	0%	0%	0.000	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw08	0	0	0%	0%	0.000	5	5	21%	1%	0.000	5	5	21%	1%	0.000
nw15	0	0	0%	0%	0.000	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw21	0	0	0%	0%	0.000	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw22	0	0	0%	0%	0.000	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw12	0	0	0%	0%	0.000	3	-2	11%	0%	0.000	3	-2	11%	0%	0.000
nw39	0	0	0%	0%	0.000	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw20	0	0	0%	0%	0.000	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw23	0	0	0%	0%	0.000	1	1	5%	0%	0.015	2	-4	11%	-1%	0.000
nw37	0	0	0%	0%	0.000	0	0	0%	0%	0.016	0	0	0%	0%	0.015
nw26	0	0	0%	0%	0.000	0	0	0%	0%	0.000	2	7	9%	1%	0.000
nw10	0	0	0%	0%	0.016	4	4	17%	0%	0.016	4	4	17%	0%	0.000
nw34	0	0	0%	0%	0.000	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw43	0	0	0%	0%	0.015	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw42	0	0	0%	0%	0.000	2	2	9%	0%	0.000	2	2	9%	0%	0.000
nw28	0	0	0%	0%	0.016	1	126	6%	10%	0.000	1	126	6%	10%	0.000
nw25	0	0	0%	0%	0.000	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw38	0	0	0%	0%	0.000	2	2	9%	0%	0.016	2	2	9%	0%	0.000
nw27	0	0	0%	0%	0.000	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw24	0	0	0%	0%	0.000	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw35	0	0	0%	0%	0.015	0	0	0%	0%	0.000	0	0	0%	0%	0.015
nw36	0	0	0%	0%	0.016	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw29	0	0	0%	0%	0.000	0	0	0%	0%	0.016	0	0	0%	0%	0.000
nw30	0	0	0%	0%	0.000	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw31	0	0	0%	0%	0.000	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw19	0	0	0%	0%	0.000	8	8	20%	0%	0.032	8	8	20%	0%	0.015
nw33	0	0	0%	0%	0.000	0	0	0%	0%	0.016	0	0	0%	0%	0.000
nw09	0	0	0%	0%	0.000	7	7	18%	0%	0.015	7	7	18%	0%	0.016
nw07	0	0	0%	0%	0.016	3	3	8%	0%	0.031	3	3	8%	0%	0.031
aa02	165	1242	31%	24%	0.703	235	1008	44%	19%	1.093	241	807	45%	16%	1.078
nw06	0	0	0%	0%	0.016	12	12	24%	0%	0.078	12	12	24%	0%	0.078
aa04	83	995	19%	14%	0.562	132	801	31%	11%	0.938	138	491	32%	7%	1.000
aa06	135	1139	21%	16%	0.969	255	1016	39%	14%	1.953	269	528	42%	7%	2.204
kl01	8	863	15%	12%	0.093	8	863	15%	12%	0.094	8	863	15%	12%	0.094
aa05	256	1877	32%	23%	2.094	406	1638	51%	20%	3.484	432	756	54%	9%	3.844
aa03	259	1577	31%	18%	2.266	413	1453	50%	17%	3.797	428	924	52%	11%	3.891
nw11	0	0	0%	0%	0.016	10	10	26%	0%	0.125	11	11	28%	0%	0.172
aa01	203	1217	25%	14%	2.032	330	967	40%	11%	3.438	343	403	42%	5%	3.641
nw18	0	0	0%	0%	0.016	47	35	38%	0%	0.750	49	-49	40%	0%	0.828
us02	55	2844	55%	21%	0.984	57	3605	57%	26%	1.297	57	3605	57%	26%	1.297
nw13	0	0	0%	0%	0.016	2	2	4%	0%	0.078	2	2	4%	0%	0.078
us04	62	18576	38%	66%	1.890	93	20409	57%	73%	3.031	93	20409	57%	73%	3.031
kl02	2	0	3%	0%	0.203	2	0	3%	0%	0.204	2	0	3%	0%	0.203
nw03	0	0	0%	0%	0.093	6	6	10%	0%	0.500	6	6	10%	0%	0.578
nw01	0	0	0%	0%	0.125	3	-48	2%	0%	0.671	3	-48	2%	0%	0.687
us03	27	36780	35%	43%	7.453	27	36780	35%	43%	7.609	27	36780	35%	43%	7.546
nw04	0	0	0%	0%	0.187	1	1	3%	0%	0.453	1	1	3%	0%	0.437
nw02	0	0	0%	0%	0.234	2	-226	1%	0%	0.953	2	-226	1%	0%	0.937
nw17	0	0	0%	0%	0.329	7	7	11%	0%	1.641	7	7	11%	0%	1.609
nw14	0	0	0%	0%	0.313	4	4	5%	0%	1.485	5	5	7%	0%	15.281
nw16	0	0	0%	0%	0.469	7	7	5%	0%	2.250	8	8	6%	0%	4.078
nw05	0	0	0%	0%	0.844	10	10	14%	0%	5.109	10	10	14%	0%	5.079
us01	59	53729	41%	5%	161.234	59	53729	41%	5%	160.297	59	53729	41%	5%	160.781

Table 7: Added value of clique & row combination techniques over 2-rowset technique

Problem	Results 2-rowset, equal rows, clique, equal rows					Results 2-rowset, equal rows, row combinations				
	CS	SS	%CS	%SS	Time (s)	CS	SS	%CS	%SS	Time (s)
nw41	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw32	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw40	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw08	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw15	0	58	0%	12%	0.000	0	0	0%	0%	0.000
nw21	0	5	0%	1%	0.000	0	0	0%	0%	0.000
nw22	0	12	0%	2%	0.000	0	0	0%	0%	0.000
nw12	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw39	0	2	0%	0%	0.000	0	0	0%	0%	0.000
nw20	0	30	0%	4%	0.016	0	0	0%	0%	0.000
nw23	0	93	0%	13%	0.000	0	0	0%	0%	0.000
nw37	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw26	2	123	9%	16%	0.000	0	0	0%	0%	0.000
nw10	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw34	0	39	0%	4%	0.000	0	0	0%	0%	0.000
nw43	0	0	0%	0%	0.015	0	0	0%	0%	0.000
nw42	0	118	0%	11%	0.000	0	0	0%	0%	0.000
nw28	0	384	0%	32%	0.031	0	0	0%	0%	0.015
nw25	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw38	0	232	0%	19%	0.016	0	0	0%	0%	0.000
nw27	0	137	0%	10%	0.000	0	0	0%	0%	0.000
nw24	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw35	0	256	0%	15%	0.000	0	0	0%	0%	0.000
nw36	0	265	0%	15%	0.016	0	0	0%	0%	0.000
nw29	0	0	0%	0%	0.015	0	0	0%	0%	0.000
nw30	0	10	0%	0%	0.000	0	0	0%	0%	0.016
nw31	0	125	0%	5%	0.016	0	0	0%	0%	0.000
nw19	0	0	0%	0%	0.062	0	0	0%	0%	0.000
nw33	0	112	0%	4%	0.000	0	0	0%	0%	0.000
nw09	0	0	0%	0%	0.000	0	0	0%	0%	0.000
nw07	0	0	0%	0%	0.016	0	0	0%	0%	0.016
aa02	164	1328	31%	26%	0.329	165	1242	31%	24%	0.203
nw06	0	0	0%	0%	0.016	0	0	0%	0%	0.000
aa04	83	1072	19%	15%	0.281	83	995	19%	14%	0.156
aa06	137	1347	21%	18%	0.469	135	1139	21%	16%	0.234
kl01	8	905	15%	12%	0.063	8	863	15%	12%	0.031
aa05	252	1944	31%	23%	0.688	256	1877	32%	23%	0.547
aa03	260	1857	32%	22%	0.719	259	1577	31%	18%	0.516
nw11	0	0	0%	0%	0.016	0	0	0%	0%	0.016
aa01	205	1353	25%	15%	0.781	203	1217	25%	14%	0.485
nw18	0	2	0%	0%	0.078	0	0	0%	0%	0.031
us02	55	6686	55%	49%	0.359	55	2844	55%	21%	0.172
nw13	0	0	0%	0%	0.063	0	0	0%	0%	0.031
us04	56	17542	34%	63%	0.515	62	18576	38%	66%	0.625
kl02	2	0	3%	0%	0.406	2	0	3%	0%	0.109
nw03	0	0	0%	0%	0.234	0	0	0%	0%	0.109
nw01	0	168	0%	0%	1.016	0	0	0%	0%	0.125
us03	26	41851	34%	49%	3.891	27	36780	35%	43%	2.062
nw04	0	0	0%	0%	0.515	0	0	0%	0%	0.187
nw02	0	2	0%	0%	2.391	0	0	0%	0%	0.234
nw17	0	0	0%	0%	0.843	0	0	0%	0%	0.359
nw14	0	0	0%	0%	0.922	0	0	0%	0%	0.328
nw16	0	0	0%	0%	3.594	0	0	0%	0%	0.781
nw05	0	0	0%	0%	2.968	0	0	0%	0%	0.890
us01	59	86201	41%	8%	167.219	59	53729	41%	5%	13.234

Table 8: Results of applying different sequences of preprocessing techniques

	Sequence 1	Sequence 2	Sequence 3	Sequence 4	Sequence 5	Sequence 6
Number of instances	54	54	54	54	54	54
Number of instances with subset reduction	54	54	54	54	54	54
Number of instances with constraint reduction	12	36	12	12	12	36
Largest % subset reduction	60%	75%	65%	60%	65%	75%
Largest % constraint reduction	55%	57%	55%	55%	55%	57%
Smallest % subset reduction	3%	3%	3%	3%	3%	3%
Smallest % constraint reduction	0%	0%	0%	0%	0%	0%
Average % subset reduction	25%	25%	26%	25%	26%	26%
Average % constraint reduction	5%	14%	6%	5%	6%	13%
Largest time benefit Cplex solver (sec)	10.59	25.27	13.27	10.55	15.45	16.21
Largest % time benefit Cplex solver	80%	63%	100%	67%	80%	68%
Average time benefit Cplex solver (sec)	0.46	1.12	0.83	0.15	0.97	0.64
Average % time benefit Cplex solver	26%	23%	32%	31%	28%	20%
Total % time benefit Cplex solver	11%	27%	20%	4%	24%	16%