

Information-based branching schemes for binary linear mixed integer problems

Fatma Kılınç Karzan · George L. Nemhauser ·
Martin W. P. Savelsbergh

Received: 23 March 2009 / Accepted: 29 November 2009 / Published online: 13 January 2010
© Springer and Mathematical Programming Society 2009

Abstract Branching variable selection can greatly affect the effectiveness and efficiency of a branch-and-bound algorithm. Traditional approaches to branching variable selection rely on estimating the effect of the candidate variables on the objective function. We propose an approach which is empowered by exploiting the information contained in a family of fathomed subproblems, collected beforehand from an incomplete branch-and-bound tree. In particular, we use this information to define new branching rules that reduce the risk of incurring inappropriate branchings. We provide computational results that demonstrate the effectiveness of the new branching rules on various benchmark instances.

Keywords Branch-and-bound · Variable selection · Computational algorithms

Mathematics Subject Classification (2000) 90C11 Mixed integer programming · 90B40 Search theory · 90-08 Computational methods · 90-04 Explicit machine computation and programs (not the theory of computation or programming)

1 Introduction

Branch-and-bound (B&B) empowered with advanced features such as presolve, cuts, heuristics, and strong branching is the preferred algorithm for solving Mixed-Integer Linear Programming (MILP) problems. Branching, that is, partitioning the feasible

F. Kılınç Karzan (✉) · G. L. Nemhauser · M. W. P. Savelsbergh
H. Milton Stewart School of Industrial and Systems Engineering,
Georgia Institute of Technology, 765 Ferst Drive, NW, Atlanta, GA 30332-0205, USA
e-mail: fkilinc@gatech.edu

G. L. Nemhauser
e-mail: gnemhaus@isye.gatech.edu

M. W. P. Savelsbergh
e-mail: mwps@isye.gatech.edu

region of a MILP into two or more subproblems, is at the heart of all B&B algorithms and thus the importance of good branching strategies was recognized early in the development of MILP algorithms [11]. A subproblem corresponding to a node in a B&B tree is *fathomed* when we are able to certify that it has been fully explored, that is, all of its feasible solutions have been explicitly or implicitly visited; otherwise, it's referred to as an *open* subproblem. The ultimate theoretical goal in branching is to find a strategy that minimizes the total number of B&B nodes that need to be evaluated. Typically the feasible region is divided by branching on a variable, that is, identifying an integer variable x_j which has a fractional value \bar{x}_j at the current node and creating two new subproblems by enforcing $x_j \leq \lfloor \bar{x}_j \rfloor$ in one of them and $x_j \geq \lceil \bar{x}_j \rceil$ in the other. Thus, every open subproblem is restricted to be identical to a previously examined subproblem except for its chronologically last branched variable. Therefore, inappropriate branchings performed at the beginning can jeopardize the effectiveness of B&B [20].

Changing the rule for branching variable selection can have a dramatic effect on the overall time needed to solve a problem. With only a few exceptions, existing rules for branching variable selection are based on the impact of a candidate variable on the objective function values of the Linear Programming (LP) relaxation in the child nodes. The candidate variable having the greatest estimated impact is chosen for branching. The motivation for such a rule for branching variable selection is that maximizing the degradation of the objective function values of the LP relaxation at the child nodes gives tighter bounds on the open nodes. For a comprehensive study of early rules for branching variable selection see [29]. *Pseudocosts*, introduced by Bénichou et al. [11], estimate the degradation in the objective function value of the LP relaxation of the child node per unit change in the value of a candidate branching variable. *Strong branching*, introduced by Applegate et al. [8] to solve large-scale traveling salesman problems, tentatively branches on a candidate branching variable and performs a fixed, limited number of dual simplex pivots to get a lower bound on the degradation of the objective function value of the LP relaxation of the child node. Achterberg et al. [5] examine various combinations of these rules and suggest *reliability branching*, a hybrid between pseudocost and strong branching in which strong branching is performed for candidate variables whose pseudocost values have been updated fewer than a fixed number of times. Glankwamdee and Linderoth [22] suggest *lookahead branching*, which estimates and uses the impact of a branching decision on the values of the objective function of the LP relaxation of the child nodes two levels deeper than the current node. Gilpin and Sandholm [21] propose branching based on the entropy (remaining uncertainty) of a variable. The entropy is estimated by treating the fractional portions of integer variables in the LP solution as probabilities, indicating the probability with which the variable is expected to be greater than its current value in the optimal solution. Patel and Chinneck [37] show that one can reach feasibility for MILPs much faster by selecting the branching variable based on its impact on the active constraints in the parent LP relaxation, instead of using the traditional approach of selecting the branching variable based on its impact on the objective function.

Historically, variable disjunctions have been preferred as a mechanism for partitioning the solution space due to the computational benefits of the dual simplex algorithm

when solving the resulting subproblems. However, the partitioning of the solution space can be based on any disjunction. Beale and Tomlin [10] suggest branching on disjunctions determined by “special ordered sets.” More recently, partitioning the solution space using general disjunctions has been investigated (see [15,27,30,31,36]).

The information-based variable selection methods we propose are quite different from the existing approaches as they rely on collecting information up-front (as opposed to during B&B) and they estimate the impact of a candidate branching variable on the fathoming decision of child nodes (as opposed to the objective function value of the LP relaxation of child nodes). The methods recognize that the branching decisions made at the top of the B&B tree are the most important ones and use two ideas:

1. A *restart* strategy. To make more informed decisions at the top of the B&B tree, we perform an incomplete B&B search and collect information on fathomed nodes and exploit this information in the actual B&B search.
2. Fathoming-based branching variable selection. To generate small B&B trees, we choose branching variables that are likely to lead to fathoming of child nodes.

Note that using information derived from nodes that are fathomed early in the search, as in back-jumping or learning B&B (see [32,40]), may be ineffective as this information may already be tainted by inappropriate branchings. Therefore, in our approach, we strengthen the information on fathomed subproblems by eliminating unnecessary branching decisions that had no effect on their fathoming. In addition to using the derived information to guide branching decisions, we use the information to generate valid inequalities and to fix variables.

The concepts of learning and restarts have been used in artificial intelligence to solve, for example, satisfiability (SAT) problems (see [24,32]). We use some of the ideas from the SAT domain in a B&B MILP solver. As such, this work is part of an increasing effort in the integration of approaches from mathematical logic and mathematical programming (see [1–4,9,17,18,25,26,38]).

Chvátal [13] also suggests obtaining a minimal cardinality clause for each fathomed node, but this clause information is used in resolution search which is quite different from B&B.

Since Achterberg, Davey et al. [1,17] and Sandholm and Shields [38] investigate obtaining clauses from fathomed nodes, although in a heuristic fashion and without employing restarts, these papers are most closely related to our approach.

More recently, Achterberg [2,3] proposed *inference branching*, inspired by SAT solvers, in which inference values, i.e., the number of domain reductions triggered by a branching decision, are computed and used to guide branching decisions. In order to initialize inference values, he suggests collecting historical information in a fashion similar to pseudocosts and using an implication graph and clique tables. Compared to our branching rules, this idea is of the “one-step look-a-head” type as it only considers and counts the immediate propagations, whereas our rules are based on the entire active segment of a clause and looks for the overall probability of fathoming a node based on the information obtained in the collection phase. Motivated, in part, by a conference presentation of our results [28], Achterberg and Berthold [4] suggest *hybrid branching*, a strategy that combines pseudocosts, inference values, and conflict

values (related to the *variable independent decaying sum* strategy of SAT solvers [35]) with clause length information.

Restarts have become a standard component of modern SAT solvers. The most common form of a restart in SAT solvers is to restart whenever a certain number of fathomed nodes has been reached. Recently, restarts have been introduced in integer programming solvers as well. The latest releases of CPLEX (11.1), MINTO (3.1) [16,33] and SCIP (1.1.0) [39] include an automatic restart at the root node if a certain amount of preprocessing reductions is achieved. Moreover, SCIP (1.1.0) [39] offers the possibility of a restart based on a specified set of criteria, including the number of clauses generated.

The main contribution of our research is assessing the value of restarts in a binary MILP framework using clause information obtained from fathomed nodes. To strengthen the information obtained from the incomplete B&B search, we develop a MILP that derives a minimum cardinality clause for each fathomed node. (An extension of that model can be used to generate clauses from scratch.) We also provide various branching alternatives that use this information in the restart framework. Our computational study shows that the restart framework is effective for moderate and large-size instances as node counts and solution times are reduced significantly.

The remainder of the paper is organized as follows. In Sect. 2, we present some basic concepts and results. Our approach to using restart with advanced branching based on information collected from the previous incomplete tree is detailed in Sect. 3. In Sect. 3.1, we present a model to improve the information obtained from the fathomed nodes of a partial B&B tree. This model is quite general and can be used to generate generalized cover inequalities for binary MILP problems. Our computational results are provided in Sect. 4. We give conclusions and further research directions in Sect. 5.

2 Preliminaries and notation

We consider the binary MILP problem

$$\min \left\{ c^T x + d^T y \mid Ax + By \geq b, x \in \{0, 1\}^n, y \in \mathbb{R}_+^k \right\} \quad (\text{P})$$

where $c \in \mathbb{R}^n$, $d \in \mathbb{R}^k$, $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{m \times k}$. Its LP relaxation is obtained by replacing $x_i \in \{0, 1\}$ with $0 \leq x_i \leq 1$ for $i = 1, \dots, n$.

Recall that in a B&B tree, or for short a tree, a node can be fathomed in three ways: (i) the node LP-relaxation is infeasible, (ii) the optimal solution to the node LP-relaxation is integer, and (iii) the optimum value of the node LP-relaxation is no better than the objective value of the currently best known integer feasible solution. To formalize the notion of fathoming, we use the following definitions and notation.

Definition 1 Let v^* be an upper bound on the objective value of the optimum solution to (P). A partial assignment $C = (C^0, C^1)$ is called a clause if

$$\left\{ (x, y) \in [0, 1]^n \times \mathbb{R}_+^k : c^T x + d^T y \leq v^*, \quad Ax + By \geq b, \right. \\ \left. x_j = 0 \quad \forall j \in C^0, \quad x_j = 1 \quad \forall j \in C^1 \right\}$$

is empty.

Let N_0 denote the root node, N_i denote a node in the tree and $C_i = (C_i^0, C_i^1)$ be the set of binary variables fixed at node N_i , where C_i^0 (C_i^1) denotes the indices of binary variables fixed to 0 (1). Without loss of generality, we can assume that $C_0 = \emptyset$. Therefore for any given fathomed node N_i in a B&B tree, the corresponding pair of index sets (C_i^0, C_i^1) , gives a clause. Let f_j^l denote *fixing* the binary variable x_j to the value $l \in \{0, 1\}$, i.e. setting $x_j = l$.

Definition 2 A clause C is called minimal if for all $f_j^l \in C$, the partial assignment $C \setminus \{f_j^l\}$ is not a clause.

If C is a minimal clause, no node N_i in the tree with $C_i \subset C$ can be fathomed by any of the fathoming rules. Moreover any node N_i with $C_i \supseteq C$ can be fathomed together with the subtree rooted at N_i . Note that there exists a minimal clause (not necessarily unique) associated with each fathomed node in any binary B&B tree.

Given a clause $C = (C^0, C^1)$, the inequality

$$\sum_{i \in C^0} x_i + \sum_{i \in C^1} (1 - x_i) \geq 1 \tag{1}$$

eliminates all solutions that do not comply with it. Clearly (1) might cut off some feasible solutions. However, the region cut off by (1) is guaranteed not to contain any feasible solution with an objective value better than the current best objective value, v^* . Furthermore (1) is a generalized cover inequality and hence, by obtaining minimal clauses, we actually derive a generalized cover inequality for (P).

Since shorter clauses yield fathoming more quickly, we are particularly interested in finding a minimal clause of *minimum cardinality*. Identifying a minimum cardinality clause is closely related to identifying a minimally infeasible subsystem of inequalities (IIS) [7,23]. Achterberg [1] shows that the minimal cardinality bound IIS problem is NP-Hard, which implies the following result:

Corollary 1 *Finding a minimum cardinality clause associated with a fathomed node in a tree is NP-Hard.*

As opposed to the heuristic approach for reducing the size of the clauses in Achterberg [1], we choose to obtain a minimum cardinality clause for each clause corresponding to a fathomed node.

3 An information-based approach

We wish to efficiently identify the most useful clauses and use them effectively in B&B in a restart framework. Our approach is based on deriving information in the

form of clauses from the fathomed nodes of an incomplete tree. We employ this information in guiding the search through designing advanced propagation and branching schemes as well as in generating valid inequalities of the form (1).

It is quite likely that the clauses obtained from an incomplete tree are not minimal. We therefore strengthen the information on fathomed nodes by identifying the branching decisions that are essential to the fathoming of the node. We do this by solving a MILP model that obtains a minimum cardinality clause from a given basic clause.

3.1 Identifying minimum cardinality clauses

We present a model that can be used to generate a clause of minimum cardinality. Without loss of generality, we assume that the upper and lower bounds of the binary variables are included in the original formulation as constraints, i.e., $Ax + By \geq b$ includes $0 \leq x_i \leq 1 \forall i \in \{1, \dots, n\}$. Let v^* be an upper bound on the objective value of the optimum solution to (P). We can fathom any node of the tree which is either infeasible or has an objective value greater than v^* . Fathoming based on the integrality of the LP solution is quite infrequent and given v^* , we can simply fathom all nodes with objective value greater than or equal to v^* by bound, which includes fathoming of all integer solutions that are no better than the current best bound as well.

Consider a leaf node of the tree that is fathomed by bound and denote the corresponding set of variable fixings by $C = (C^0, C^1)$. The LP relaxation of this leaf node is

$$\min c^T x + d^T y \tag{2}$$

$$\text{s.t. } Ax + By \geq b \tag{3}$$

$$x_i \geq 1 \quad \forall i \in C^1 \tag{4}$$

$$-x_i \geq 0 \quad \forall i \in C^0 \tag{5}$$

$$x \in \mathbb{R}_+^n, y \in \mathbb{R}_+^k \tag{6}$$

Define the following variables z_i^0 (z_i^1) for $i \in C^0$ ($i \in C^1$):

$$z_i^0 = \begin{cases} 1, & \text{if inequality } -x_i \geq 0 \text{ is added to the LP relaxation;} \\ 0, & \text{otherwise,} \end{cases}$$

and

$$z_i^1 = \begin{cases} 1, & \text{if inequality } x_i \geq 1 \text{ is added to the LP relaxation;} \\ 0, & \text{otherwise.} \end{cases}$$

Using these new variables, the Mixed Integer Quadratically Constrained Program (MIQCP)

$$\min c^T x + d^T y \tag{7}$$

$$\text{s.t. } Ax + By \geq b \tag{8}$$

$$z_i^1 x_i \geq z_i^1 \quad \forall i \in C^1 \tag{9}$$

$$-z_i^0 x_i \geq 0 \quad \forall i \in C^0 \tag{10}$$

$$x \in \mathbb{R}_+^n, y \in \mathbb{R}_+^k \tag{11}$$

$$z_i^0 \in \{0, 1\} \quad \forall i \in C^0, \quad z_i^1 \in \{0, 1\} \quad \forall i \in C^1. \tag{12}$$

is equivalent to the node LP when all of the binaries in it are set to 1.

We want to find a minimum number of bound inequalities such that their inclusion in the original linear programming relaxation of (P) will still lead to a fathoming, i.e., either the LP relaxation is infeasible or the objective value exceeds the lower bound value, v^* . Since the original root LP relaxation is assumed to be feasible and bounded from below, its dual is also feasible. Thus we know that in the case of infeasibility of the node LP, the dual of the node LP is unbounded since when we add new bound inequalities to the primal, we are just adding new columns to the dual of the node LP. Therefore when a node is fathomed by infeasibility, we will be able to find a dual solution with an objective value strictly greater than v^* . So now we consider the dual of the above formulation by treating the variables z_i^0 and z_i^1 as parameters and we obtain

$$\max \lambda^T b + \sum_{i \in C^1} \gamma_i^1 z_i^1 \tag{13}$$

$$\text{s.t. } \lambda^T A_i + \gamma_i^1 z_i^1 \leq c_i \quad \forall i \in C^1 \tag{14}$$

$$\lambda^T A_i - \gamma_i^0 z_i^0 \leq c_i \quad \forall i \in C^0 \tag{15}$$

$$\lambda^T A_i \leq c_i \quad \forall i \in \{1, \dots, n\} \setminus (C^0 \cup C^1) \tag{16}$$

$$\lambda^T B_j \leq d_j \quad \forall j \in \{1, \dots, k\} \tag{17}$$

$$\lambda_l \geq 0 \quad \forall l \in \{1, \dots, m\} \tag{18}$$

$$\gamma_i^0 \geq 0 \quad \forall i \in C^0, \quad \gamma_i^1 \geq 0 \quad \forall i \in C^1. \tag{19}$$

Since we are only interested in the existence of dual solutions with objective value greater than or equal to v^* , we can equivalently turn the objective into a constraint. By considering z_i^0 and z_i^1 as binary variables with the condition that at most one of them can be set to 1 for each i (since $C^0 \cap C^1 = \emptyset$, we don't need to include these constraints explicitly), we obtain the following MIQCP where we minimize the sum of z_i^0 and z_i^1 :

$$\min \sum_{i \in C^0} z_i^0 + \sum_{i \in C^1} z_i^1 \tag{20}$$

$$\text{s.t. } \lambda^T b + \sum_{i \in C^1} \gamma_i^1 z_i^1 \geq v^* \tag{21}$$

$$\lambda^T A_i + \gamma_i^1 z_i^1 \leq c_i \quad \forall i \in C^1 \tag{22}$$

$$\lambda^T A_i - \gamma_i^0 z_i^0 \leq c_i \quad \forall i \in C^0 \tag{23}$$

$$\lambda^T A_i \leq c_i \quad \forall i \in \{1, \dots, n\} \setminus (C^0 \cup C^1) \tag{24}$$

$$\lambda^T B_j \leq d_j \quad \forall j \in \{1, \dots, k\} \tag{25}$$

$$\lambda_l \geq 0 \quad \forall l \in \{1, \dots, m\} \tag{26}$$

$$z_i^0 \in \{0, 1\}, \quad \gamma_i^0 \geq 0 \quad \forall i \in C^0 \tag{27}$$

$$z_i^1 \in \{0, 1\}, \quad \gamma_i^1 \geq 0 \quad \forall i \in C^1. \tag{28}$$

In order to bound the dual variables γ and λ from above by 1, we introduce another variable α , we relabel $\alpha\lambda$ and $\alpha\gamma$ as λ and γ , respectively and to linearize the model, we introduce u_i^0 and u_i^1 for the nonlinear terms $\gamma_i^0 z_i^0$ and $\gamma_i^1 z_i^1$, respectively. This yields the MILP formulation:

$$\min \sum_{i \in C^0} z_i^0 + \sum_{i \in C^1} z_i^1 \tag{29}$$

$$\text{s.t. } \lambda^T b + \sum_{i \in C^1} u_i^1 - \alpha v^* \geq 0 \tag{30}$$

$$\lambda^T A_i + u_i^1 - \alpha c_i \leq 0 \quad \forall i \in C^1 \tag{31}$$

$$\lambda^T A_i - u_i^0 - \alpha c_i \leq 0 \quad \forall i \in C^0 \tag{32}$$

$$\lambda^T A_i - \alpha c_i \leq 0 \quad \forall i \in \{1, \dots, n\} \setminus (C^0 \cup C^1) \tag{33}$$

$$\lambda^T B_j - \alpha d_j \leq 0 \quad \forall j \in \{1, \dots, k\} \tag{34}$$

$$u_i^0 \leq \gamma_i^0, \quad u_i^0 \leq z_i^0, \quad u_i^0 - \gamma_i^0 - z_i^0 \geq -1 \quad \forall i \in C^0 \tag{35}$$

$$u_i^1 \leq \gamma_i^1, \quad u_i^1 \leq z_i^1, \quad u_i^1 - \gamma_i^1 - z_i^1 \geq -1 \quad \forall i \in C^1 \tag{36}$$

$$0 \leq \lambda_l \leq 1 \quad \forall l \in \{1, \dots, m\} \tag{37}$$

$$z_i^0 \in \{0, 1\}, \quad 0 \leq u_i^0, \gamma_i^0 \leq 1 \quad \forall i \in C^0 \tag{38}$$

$$z_i^1 \in \{0, 1\}, \quad 0 \leq u_i^1, \gamma_i^1 \leq 1 \quad \forall i \in C^1 \tag{39}$$

$$0 < \alpha. \tag{40}$$

Note that the fixing of the last variable on the path from the root node to a leaf node is required to fathom the leaf node. Hence in any feasible solution to the above formulation, we will always have the corresponding binary variable (z_i^0 or z_i^1) fixed to 1. In our computations, we take advantage of this fact by actually fixing the value

of z_i corresponding to the last branched variable to 1. We also replace $\alpha > 0$ with $\alpha \geq 10^{-5}$, which is equivalent to a big- M formulation where the big- M value can be chosen as $\frac{1}{\alpha} = 10^5$. We are aware of the fact that this lower bound on α value might lead to non-optimal solutions for some instances, i.e. for some instances using a smaller bound will lead to a clause with smaller cardinality. A more careful analysis using the inverses of the maximum upper bound values for the dual variables can be applied to find better bounds in this regard. However, in order to avoid possible numerical difficulties, we simply choose to use 10^{-5} . This issue can also be handled using the indicator-variable feature provided in some integer programming solvers.

As an aside, an auxiliary problem can be solved to obtain a clause (i.e., a set of binary variable fixings that leads to a fathoming) without any knowledge of a fathomed node. Furthermore, when v^* is the optimal objective value, the optimal value to this problem will give a lower bound on the length of any path in the B&B tree. This problem is given by:

$$\min \sum_{i=1}^n (z_i^0 + z_i^1) \tag{41}$$

$$\text{s.t. } \lambda^T b + \sum_{i=1}^n u_i^1 - \alpha v^* \geq 0 \tag{42}$$

$$\lambda^T A_i + u_i^1 - u_i^0 - \alpha c_i \leq 0 \quad \forall i \in \{1, \dots, n\} \tag{43}$$

$$\lambda^T B_j - \alpha d_j \leq 0 \quad \forall j \in \{1, \dots, k\} \tag{44}$$

$$u_i^0 \leq \gamma_i^0, \quad u_i^0 \leq z_i^0, \quad u_i^0 - \gamma_i^0 - z_i^0 \geq -1 \quad \forall i \in \{1, \dots, n\} \tag{45}$$

$$u_i^1 \leq \gamma_i^1, \quad u_i^1 \leq z_i^1, \quad u_i^1 - \gamma_i^1 - z_i^1 \geq -1 \quad \forall i \in \{1, \dots, n\} \tag{46}$$

$$z_i^0 + z_i^1 \leq 1 \quad \forall i \in \{1, \dots, n\} \tag{47}$$

$$0 \leq \lambda_l \leq 1 \quad \forall l \in \{1, \dots, m\} \tag{48}$$

$$z_i^0, z_i^1 \in \{0, 1\}, \quad 0 \leq u_i^0, u_i^1, \gamma_i^0, \gamma_i^1 \leq 1 \quad \forall i \in \{1, \dots, n\} \tag{49}$$

$$0 < \alpha. \tag{50}$$

If an upper bound on the objective value is not available, then we fathom a node only when the corresponding LP relaxation is infeasible or the corresponding optimum solution is integral. In such a case, our models can still be used to identify clauses corresponding to infeasible node LPs by setting the objective function to zero, i.e. $c = 0, d = 0$ and $v^* = 1$.

3.2 Guiding search with dynamic branching

Once the clause information is collected, there is still the question of how to use this information to aid the search in the restart phase. Let $\mathcal{C} = \{C_1, \dots, C_K\}$ be a set

of clauses. Consider a node of the tree \tilde{N} other than the root node, and let $\tilde{C}^0(\tilde{C}^1)$ denote the set of binary variables fixed to 0 (1). We say a clause $C = (C^0, C^1)$ is active at \tilde{N} if $C^0 \cap \tilde{C}^1 = \emptyset$ and $C^1 \cap \tilde{C}^0 = \emptyset$, i.e. if it is possible to obtain a descendent node (not necessarily a child) from the current node that can be fathomed by the clause C . Whenever a clause becomes inactive for a particular node, it will remain inactive for all the descendent nodes of that node. Since some variables are already fixed at \tilde{N} , the active clauses can be updated using this information, i.e., the clause $C = (C^0 \setminus \tilde{C}^0, C^1 \setminus \tilde{C}^1)$ indicates the possible extension of the current node to a child node which can be fathomed by clause C . In the rest of the text whenever we refer to an active clause at a node, we actually mean the updated version of the clause.

Whenever an active clause C has only one variable, i.e., $|C| = 1$, we can immediately fix the value of that variable. Suppose $C = C^0 = \{j\}$, then we can set $x_j = 1$ and create only a single child node, as the other branch will automatically be fathomed by the clause. We refer to this as *propagation*.

Given a node and a set of active clauses at that node, there are several ways to use this information in determining a branching variable. Let \tilde{x} be the vector of current LP relaxation values of variables at the node. We first weight each active clause, denoted by $\omega(C_i)$, to estimate its importance in fathoming. We have tested four different alternatives to estimate $\omega(C_i)$:

0. $\omega(C_i) = 1$ for all clauses C_i (i.e. all clauses are of equal importance),
1. $\omega(C_i) = \frac{1}{|C_i|}$ where $|C_i|$ is the number of variables included in clause C_i (i.e. short clauses are preferred),
2. $\omega(C_i) = 2^{-|C_i|}$ (i.e. exponentially higher preference is given to the shorter clauses),
3. $\omega(C_i) = \frac{1}{\max\left\{\sum_{j \in C_i^0} \tilde{x}_j + \sum_{j \in C_i^1} (1 - \tilde{x}_j) - 1, 10^{-10}\right\}}$ where we use the closeness to violation of the clause inequality (1).

Then using the weights of the clauses, we estimate the effectiveness of fixing the binary variable x_j to 0 (1), denoted by β_j^0 (β_j^1). We have tested two alternatives to estimate the overall effect of creating a branch with $x_j = 0$ (1):

0. $\beta_j^0 = \max\{\omega(C_i) : j \in C_i^0\}$ and $\beta_j^1 = \max\{\omega(C_i) : j \in C_i^1\}$,
1. $\beta_j^0 = \sum_{i: j \in C_i^0} \omega(C_i)$ and $\beta_j^1 = \sum_{i: j \in C_i^1} \omega(C_i)$.

Let β_j denote the overall effect of branching on variable x_j . To estimate β_j , we combine β_j^0 and β_j^1 using rules that have been derived to combine pseudocosts (see [29]). We suggest and test the following alternatives:

0. $\beta_j = \min\{\tilde{x}_j, 1 - \tilde{x}_j\} \cdot (\beta_j^0 + \beta_j^1)$,
1. $\beta_j = \beta_j^0 + \beta_j^1$,
2. $\beta_j = \max\{\beta_j^0, \beta_j^1\} + 10 \cdot \min\{\beta_j^0, \beta_j^1\}$,
3. $\beta_j = \max\{\beta_j^0, 10^{-6}\} \cdot \max\{\beta_j^1, 10^{-6}\}$.

Note that the first alternative considers the fractionality of the variable, whereas the second one simply adds the individual effects, the third one is similar to the weights

used in strong branching and the last one is a multiplicative rule inspired by the one in Achterberg [2].

We denote a specific rule for determining a branching variable as $a - b - c$ where $a \in \{0, 1, 2, 3\}$, $b \in \{0, 1\}$ and $c \in \{0, 1, 2, 3\}$.

4 Computational results

The purpose of our computational study is to assess the benefits of learning and restart with both *basic* information directly obtained from the incomplete tree and *improved* information obtained through the minimum clause identification models. We investigate the impact of propagation, of branching, and of adding clause inequalities (1).

4.1 Experimental setup

All experiments are done on a Dual 2.4GHz Xeon Linux workstation with 2GB of RAM. CPLEX 11.1 [16] is used both as a representative state-of-the-art commercial MILP solver with which we compare our new methods, and as the basic MILP solver framework upon which our new methods are built. A C++ program manages the interface between CPLEX and our new routines, especially the process of obtaining necessary information via the callback routines in the CPLEX callable library.

In all of the experiments, including the information collection and restart phases, whenever the branching rule is not overwritten by one of our rules, we use strong branching.

In order to make the comparison among the different branching rules as fair as possible, we provide the value of the optimal solution as a cutoff value. Thus in all of the restart experiments, node counts and solution times indicate the effort required to prove optimality. As heuristics have no effect in this setting, they are turned off. Advanced features such as presolve and cut generation are applied at the root node. The root node relaxation thus obtained is used in the minimum clause identification model. We allow CPLEX to perform local preprocessing and cut generation at nodes both in the information collection and restart phases. Although including local information in the auxiliary problems for obtaining minimum cardinality clauses may lead to significant improvements, we still only use the root node LP to construct the auxiliary problems because retrieving local information is cumbersome and can be very time consuming. Moreover, in the clause improvement phase, we provide the value of the optimal solution as v^* and use default CPLEX to solve the minimum clause identification models while enforcing a time limit of 5 s for each solve.

To collect information from a diverse set of nodes in the B&B tree, we use best-first search in the information collection phase. As we provide the optimum value to the problems in all of the experiments, the node selection strategy does not have a significant effect on the number of nodes. As the easy retrieval of the basis provides some benefits in terms of time, depth-first search is used in the restart solves as well as comparison runs with CPLEX B&B and CPLEX dynamic search (DS).

In the information collection phase, we terminate the search based on a fixed number of fathomed nodes detected. We use a limit of 200 fathomed nodes as a base, but also investigate the impact of this limit.

We use instances from MIPLIB 3.0 [12], MIBLIB 2003 [6], and Cor@1 [14] libraries and from the test set of Mittelmann [34] for our computational experiments. In order to give results on a meaningful set of problems, we eliminate all instances that meet at least one of the following criteria:

- the instance cannot be solved with CPLEX B&B under these settings in 7,200 s (the instance is too hard);
- the instance solves in less than 60 s or the B&B tree has fewer than 1,000 nodes with CPLEX B&B under these settings (the instance is too easy);
- the clause collection phase terminates with fewer than 200 clauses (note that many instances that might have been eliminated by this criterion would have already been eliminated under the “too easy” criterion. However, the main reason for eliminating instances by this rule that were neither too hard nor too easy is that the instance had general integer variables as well as binary variables and we could not obtain 200 fathomed clauses that only contained binary variables.);
- the clause collection phase for the instance cannot be completed in a time limit of 7,200 s or due to memory limits. (Note that the time spent on the collection phase strictly depends on the instance as the collection process is halted whenever we collect 200 clauses or the whole B&B process is finished.)

In addition to this, we divide the set of instances into three groups based on their CPLEX B&B node counts as *easy*, *medium* and *hard* where *easy* instances require between 1,000 and 5,000 nodes, *medium* instances require between 5,000 and 50,000 nodes, and *hard* ones require at least 50,000 nodes. This yields 13 *hard*, 23 *medium* and 15 *easy* instances with a total of 51 instances. We present and discuss summary results for instances from these three groups in the main text. (Detailed results for all of the instances are given in the Appendix.) In the summary tables, we provide summary statistics, i.e. arithmetic and shifted geometric means with a shift of 1, for all of the instances as well as the instances within each group. For a set of nonnegative values $T = \{t_1, \dots, t_n\}$, the shifted geometric mean with a shift of δ , is given by the formula $(\prod_{i=1}^n (t_i + \delta))^{1/n} - \delta$.

In addition, we provide various performance profiles (see [19] for more information on performance profiles) to summarize our findings on the behavior of the algorithms in the restart phase. In a performance profile for the number of nodes (solution time), for a given method f and a given point (θ, γ) on the plot of f , the value γ indicates the fraction γ of the instances that were solved by that particular method f , within θ times the number of nodes (time) required by the method with fewest nodes (fastest method) for each instance. In particular, the intercepts of the plot (if any) with the vertical axis to the left indicate the fraction of the instances for which the method performed best, respectively.

4.2 The value of the restart strategy with fathoming-based branching

In Table 1, we report a summary of the computational results for our test set of instances. (Detailed results for all instances are given in Table 6 in the Appendix.) In the first part of Table 1, we provide information about the *Collection phase*, namely the number of nodes in the incomplete tree and the time (in seconds) required to

Table 1 Value of the restart strategy with fathoming-based branching

	Collection phase			Improvement phase			Restart phase			Total				
	#	Size	# (%)	Min	Max	Total	Size	Cuts&Prop&Branch		CPLEX		Cuts&Prop&Branch		
								B	I	DS	BB	B	I	
Arith.Mean	# Nodes 795.8	200	23.08	0	84.3	653.6	70.57 (35.3)	16.61	74498.6	76399.6	81421.9	104274.6	75294.4	77849
(All: 51 models)	Time 88			0.7	2.6	249.3			861.4	752.6	1026.7	1116.2	949.4	1089.9
Geo.Mean	# Nodes 691.2	200	21.07	0	13.7	56.8	24.56 (12.3)	15.14	11688.6	10095.2	11277.7	15729.7	13418.5	12804.8
(All: 51 models)	Time 41.9			0.5	1.9	86.9			494.2	417.2	524.3	677.3	581.3	712.6
Arith.Mean	# Nodes 948.3	200	26.33	0	121.4	962.1	101.15 (50.6)	16.58	265678.8	275569.9	287286.2	376527	266627.1	277480.3
(Hard: 13 models)	Time 21.3			0.1	1.3	73.6			1741	1574.7	1851.2	2166.5	1762.3	1669.6
Geo.Mean	# Nodes 866.7	200	24.8	0	35.4	183.9	77.72 (38.9)	15.37	139124.4	119371.1	125247.4	187595.9	140568.2	121816.5
(Hard: 13 models)	Time 12.8			0.1	0.9	27.1			1185.5	965.6	1083.2	1486.4	1203.8	1026
Arith.Mean	# Nodes 789	200	22.52	0	77.3	648.2	73.87 (36.9)	15.55	13540	12210.6	16108	16689.9	14329	13647.8
(Medium: 23 models)	Time 71.3			0.6	2.8	228.9			608.5	500.5	776.8	866.1	679.8	800.7
Geo.Mean	# Nodes 710.3	200	20.31	0	12	56.9	21.32 (10.7)	14.13	9465.2	7512.4	7686.5	13503.9	10685.6	9730.4
(Medium: 23 models)	Time 46.3			0.4	2.1	96.4			428.3	342.3	383.3	654.8	498.5	595.5
Arith.Mean	# Nodes 673.8	200	21.12	0	62.9	394.5	39.00 (19.5)	18.27	2278.7	2208.4	3154.1	2619.1	2952.5	3276.7
(Easy: 15 models)	Time 171.5			1.3	3.5	433			486.9	426.9	695.3	589.3	658.4	1031.4
Geo.Mean	# Nodes 544.7	200	19.34	0	7.1	20	10.86 (5.4)	16.62	1887.4	1866.3	2519.1	2318.5	2483.5	2768.3
(Easy: 15 models)	Time 97.6			0.8	2.7	200.6			288.1	272.9	451.8	360.8	391.3	684.1

construct the incomplete tree, and the number of clauses collected and their average size, i.e., the average number of binary variables involved in the clauses. In the second part, we provide information about the *Improvement phase*, namely the minimum, the maximum, and the total number of nodes as well as the corresponding solution times required for solving the minimum clause identification problems, the number of clauses improved together with the average size of the improved set of clauses. It is somewhat surprising that the easy instances took more time than the others in the collection and improvement phases. We do not have a good explanation for this. However, it is possible that although these instances are easy in the MIP sense of having small trees, they have large or hard LP relaxations.

In the third part, we provide information about the *Restart phase*, namely the node count and solution time statistics for *B-Cuts&Prop&Branch* and *I-Cuts&Prop&Branch*. In *B-Cuts&Prop&Branch* we use basic clause information, i.e., information obtained directly from the incomplete tree, whereas in *I-Cuts&Prop&Branch* we use improved clauses, i.e., information resulting from solving a minimum clause identification problem for each basic clause obtained from the incomplete tree. In both cases, clause inequalities are added to the cut pool and, whenever we have clause information on a candidate branching variable at a node, we perform branching based on the 3-1-1 combination as well as propagation. The “3-1-1” (a-b-c) choice for “weight-effect-combination” is shown to be effective in subsequent experiments.

Finally, for comparison, we provide the node counts and solution times of *CPLEX* with and without dynamic search enabled, *CPLEX-DS* and *CPLEX-BB*, respectively, as well as information on the *total* effort involved with our algorithms. For *B-Cuts&Prop&Branch* this is the sum of the efforts in collection and restart phases and for *I-Cuts&Prop&Branch* it is the sum of the efforts in collection, improvement and restart phases.

Comparing the performance of *B-Cuts&Prop&Branch* with the performance of *CPLEX B&B*, we see that the node counts and solution times improve on average (even when the statistics are added from the collection phase and restart phase and compared with that of *CPLEX B&B*). The node count improvements are substantial for some instances: for *neos-1480121* the node count goes from 1,711,005 to 484,311, and for *neos-1228986* from 137,818 to 36,581 (see Table 6 in the Appendix for details). Although the computation times improve too on average, the changes are not as significant due to the overhead of our branching strategy at each node. Note that this overhead can be reduced with advanced data structures and the use of internal *CPLEX* structures. Comparing the performance of *I-Cuts&Prop&Branch* with the performance of *B-Cuts&Prop&Branch*, we see from the performance profile of Fig. 1 that for the majority of instances the B&B trees get even smaller. This improved behavior obtained is much more significant for the instances from the *hard* set. Note that although our methods are built on top of *CPLEX B&B*, the overall performance of our approach is comparable with *CPLEX* dynamic search. By simply adding up the overall average number of nodes from the collection, improvement and restart phases for *I-Cuts&Prop&Branch* we obtain 77849.0 whereas average *CPLEX B&B* node count is 104274.6 and *CPLEX DS* node count is 81421.9. The same procedure for overall solution times leads to 1089.9s for *I-Cuts&Prop&Branch*, whereas *CPLEX B&B* is

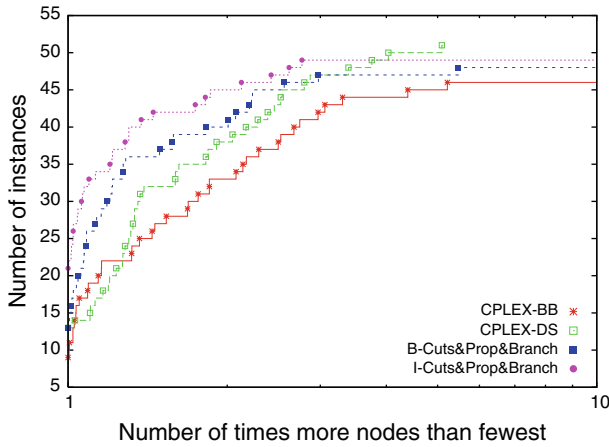


Fig. 1 Comparison of # of nodes in the restart phase

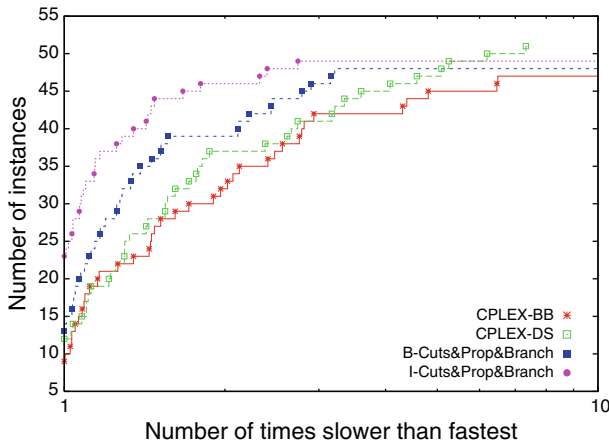


Fig. 2 Comparison of solution times in the restart phase

1116.2 s and CPLEX DS is 1026.7 s. We want to emphasize that the improvements over CPLEX are clearer when we restrict ourselves to *hard* instances.

The benefits of our restart strategy with fathoming-based branching can also be seen in Figs. 1 and 2, where we provide performance profiles in which we compare *CPLEX B&B*, *CPLEX DS*, *B-Cuts&Prop&Branch* and *I-Cuts&Prop&Branch* in terms of node counts and solution times in the restart phase.

Table 1 also shows that on average about 35% of the clauses collected are improved by solving the minimum clause identification problem, with an average reduction in size of about 28%. Yet there are instances where none of the collected clauses can be improved, e.g., *harp2*, *neos-598183* and *stein45* (see Table 6 in the Appendix). For most instances, the time to improve clauses is a fraction of the solution time of the original instance. However, some easy and medium size instances, such as *neos22*, *neos-631694*, *1rn* and *seymour-1*, are exceptions since they require quite significant improvement times, yet not so many of the clauses are improved.

Usually, this behavior is due to numerical difficulties in the solution of the LP-relaxations of the minimum clause identification problems. In fact, frequently in these cases, the LP relaxations of the minimum clause improvement problems could not be solved within the time limit of 5 s.

In Table 2, we report a summary of the computational results for our test set of instances where we focus on the contribution of each of the components of our approach. (Detailed results for all instances are given in Table 7 in the Appendix.) We report results for only adding cuts (*B-Cuts*, *I-Cuts*), only applying propagation (*B-Prop*, *I-Prop*), applying propagation and performing branching (*B-Prop&Branch*, *I-Prop&Branch*), applying propagation and adding cuts (*B-Cuts&Prop*, *I-Cuts&Prop*) and the use of all components (*B-Cuts&Prop&Branch*, *I-Cuts&Prop&Branch*).

The performance improvement from the addition of clause inequalities alone as cuts is limited, particularly when only basic clause information is used. Propagation alone gives comparable performance, although with improved clauses its performance is better than that of cuts especially on the hard instances. Adding cuts and applying propagation together gives a slight improvement to just applying propagation or cuts in both basic and improved cases. Using our branching in addition to applying propagation gives significantly greater performance. Finally, using all the components is only slightly better than just using branching and propagation. This clearly suggests that the value of adding clause inequalities as cuts is rather limited when this information is already provided in another form, as has been already reported by Achterberg [1]. The most noteworthy finding is the performance of the improved clauses. While using only the basic clauses can reduce time and node counts, the results using improved clauses together with branching and propagation are by far the most significant. We attribute this to the fact that as the clauses get shorter, they are more likely to be active at a node and their effectiveness increases.

The relative performance of the different variants of our approach (when using improved clause information) can also be seen in Figs. 3 and 4, where we provide performance profiles in terms of node counts and solution times in the restart phase. (See Figs. 5, 6 in the Appendix for the ones with basic clause information.)

Recall that we make branching and propagation decisions only when there are fractional variables in the solution to the LP relaxation at a node for which we have clause information, otherwise we let CPLEX decide on the branching variable. Table 3 provides summary statistics on the number of nodes in the tree where we actually make a decision on the branching variable, both in terms of propagation and branching. The percentages shown represent the number of times we make a branching decision as a percentage of the total number of branching decisions. (Individual results for the instances are given in Table 8 in the Appendix.) The results show that when using improved information, we make many more propagation and branching decisions. However, the number of branching decisions increases more significantly. For example, on average, nearly 50,000 branching decisions are made for the hard instances although we only collect and use 200 clauses.

4.3 Effect of collecting more information

An important question to address is how many fathomed nodes (and hence clauses) to collect from an incomplete tree. Given that collecting information from the complete

Table 2 Effect of providing information in different forms: cuts, propagation, branching

	CPLEX		Cuts		Prop		Prop&Branch		Cuts&Prop		Cuts&Prop&Branch		
	DS	BB	B	I	B	I	B	I	B	I	B	I	
Arith.Mean	#Nodes	81421.9	104274.6	98267.5	78190.6	99286	71424.2	78554.3	76915.4	96766.1	69657.1	74498.6	76399.6
(All: 51 models)	Time	1026.7	1116.2	1059.8	926.2	1158.1	949.7	877.8	740.1	1059.8	920.9	861.4	752.6
Geo.Mean	#Nodes	11277.7	15729.7	13826.9	12353.0	15774.6	13243.6	11782.9	10050.7	13971.1	12277.0	11688.6	10095.2
(All: 51 models)	Time	524.3	677.3	607.8	554.8	694.4	588.9	488.3	402.8	626.7	560.6	494.2	417.2
Arith.Mean	#Nodes	287286.2	376527	356392.8	278861.0	356427	249255.9	281874.2	277031.2	350429.3	244892.1	265678.8	275569.9
(Hard: 13 models)	Time	1851.2	2166.5	2226.8	1842.9	2241.3	1756.6	1834.8	1548.5	2177.9	1721.8	1741	1574.7
Geo.Mean	#Nodes	125247.4	187595.9	188082.9	157273.9	190045	140600.5	149230.1	121213.8	180646.8	136568.1	139124.4	119371.1
(Hard: 13 models)	Time	1083.2	1486.4	1510.9	1295.2	1523.4	1162.6	1229.7	949.6	1495.5	1153.6	1185.5	965.6
Arith.Mean	#Nodes	16108.0	16689.9	15046.4	14350.5	16563.9	15801.2	13402.4	12580.3	15070.9	14628.5	13540	12210.6
(Medium: 23 models)	Time	776.8	866.1	767.9	682.9	869.6	753.4	594.4	498.8	797.9	729.8	608.5	500.5
Geo.Mean	#Nodes	7686.5	13503.9	11602.5	10202.5	13212	11311.5	9169.7	7460.1	12146.8	10987.0	9465.2	7512.4
(Medium: 23 models)	Time	383.3	654.8	574.9	513.1	651.2	564.5	408.4	332.1	620.9	560.3	428.3	342.3
Arith.Mean	#Nodes	3154.1	2619.1	2164.7	2164.2	3271	2592.1	2243.4	2129.0	2190.7	2164.1	2278.7	2208.4
(Easy: 15 models)	Time	695.3	589.3	496	504.9	661.7	551.2	482.9	409.4	492.4	519.7	486.9	426.9
Geo.Mean	#Nodes	2519.1	2318.5	1883.1	1825.5	2393.8	2176.1	1916.3	1833.7	1883	1803.3	1887.4	1866.3
(Easy: 15 models)	Time	451.8	360.8	300.4	299.7	387.5	348.3	288.1	257.3	298.8	299.9	288.1	272.9

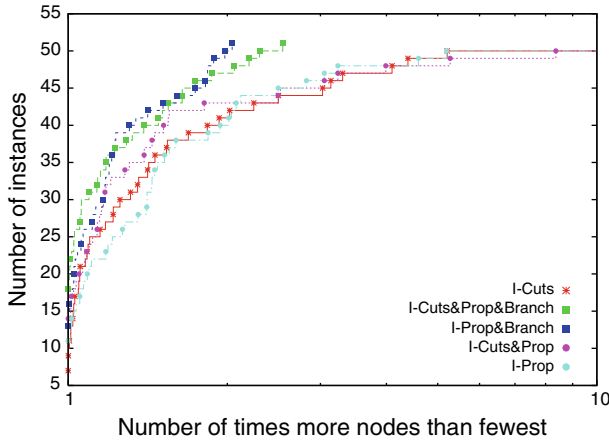


Fig. 3 Comparison of # of nodes in the restart phase for solvers with improved information

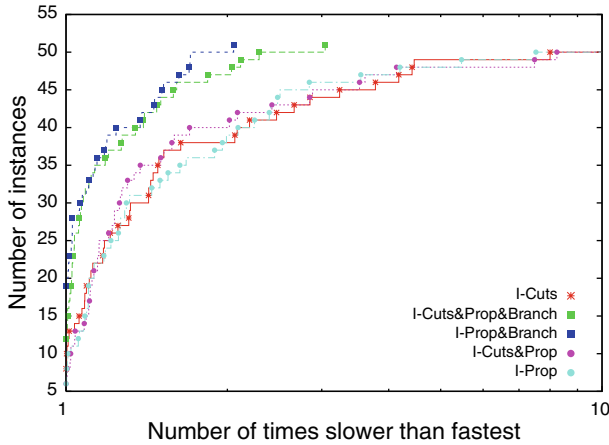


Fig. 4 Comparison of solution times in the restart phase for solvers with improved information

B&B tree is unrealistic, we did experiments with a few of the harder instances to see the effect of collecting more fathomed nodes from an incomplete tree. We varied the number of clauses collected from 10 to 1,000 and used the *I-Cuts&Prop&Branch* algorithm with the branching rule combination 3-1-1 in these experiments. The summary results are given in Table 4 and the detailed results are provided in Table 9 in the Appendix.

We observe that the number of improved clauses increases linearly with the number of collected clauses, whereas the average sizes of the collected and improved clauses do not vary much, except for *neos-1480121* and *neos14* (see Table 9 in the Appendix). As expected, collecting more information requires more nodes and time spent in the collection and improvement phases. Yet the overall value of using more information in B&B is not clear. For *neos-1228986* and *neos-863472*,

Table 3 Number of times CPLEX branching is overwritten

	# Decisions in		# Decisions in					
	<i>B-Prop</i> <i>I-Prop</i>		<i>B-Prop&Branch</i>			<i>I-Prop&Branch</i>		
	Prop	Prop	Prop	Branch	%	Prop	Branch	%
Arith.Mean(All)	91.78	1113.37	74.47	625.53	0.80	1999.16	13409.57	17.43
Geo.Mean(All)	26.94	151.36	41.67	375.12	3.18	140.90	966.61	9.62
Arith.Mean(Hard)	172.85	3828.77	96.15	994.00	0.35	7401.54	49572.54	17.89
Geo.Mean(Hard)	42.09	561.49	35.68	816.67	0.55	494.70	7304.12	6.03
Arith.Mean(Medium)	74.91	241.30	55.74	622.96	4.65	158.30	1383.83	11.00
Geo.Mean(Medium)	21.99	134.48	31.06	332.80	3.63	83.51	586.21	7.86
Arith.Mean(Easy)	47.40	97.20	84.40	310.13	13.82	139.73	507.80	23.85
Geo.Mean(Easy)	24.88	57.81	74.38	229.42	11.97	105.22	359.92	19.63

Table 4 Percentage change with respect to CPLEX B&B as the number of clauses collected increases

# Clauses	Collection		Improvement		Restart		Total (Arith.)		Total (Geo.)	
	# Nodes	Time	# Nodes	Time	# Nodes	Time	# Nodes	Time	# Nodes	Time
<i>CPLEX-BB</i>							100.00	100.00	100.00	100.00
<i>CPLEX-DS</i>							85.67	90.91	66.76	72.86
10	0.10	0.15	0.02	0.21	75.74	85.68	75.86	86.04	69.37	78.34
100	0.42	0.66	0.32	1.76	79.92	85.25	80.66	87.68	69.40	78.64
200	0.78	1.16	0.66	3.20	76.79	75.99	78.22	80.35	64.94	69.02
500	1.68	3.05	2.38	8.82	77.36	88.38	81.42	100.26	65.49	86.86
1000	3.00	7.03	4.13	17.43	74.80	96.27	81.93	120.74	66.74	104.24

there is a significant improvement, however, the impact is mixed for the rest of the instances. Note that as the amount of information increases, the time it takes to decide on a branching variable also increases because at each node we have to determine and update the active clauses, calculate the weights for each candidate variable, etc. This observation is very clear in *mas74*, where in the restart phase, the node counts for 100, 200, 500, and 1,000 clauses are roughly the same, but the solution times are significantly different.

4.4 Effect of different branching variable selection rules

Branching rules can have a significant impact on the size of the B&B tree. Table 5 summarizes the performance of the different branching rules detailed in Sect. 3.2 (see Table 10 from Appendix for detailed results on individual instances.). In these experiments, we limit the number of clauses collected to 200 and use the *I-Cuts&Prop&Branch* algorithm while varying the branching rule combination. The best rule with respect to the node count and time is shown in bold face.

Table 5 Percentage change with respect to *CPLEX B&B* for different branching rules

Rule	Restart phase				Total			
	Arith.Mean		Geo.Mean		Arith.Mean		Geo.Mean	
	# Nodes	Time	# Nodes	Time	# Nodes	Time	# Nodes	Time
<i>CPLEX-BB</i>					100.00	100.00	100.00	100.00
<i>CPLEX-DS</i>					85.67	90.91	66.76	72.86
0 0 0	116.80	110.48	76.88	76.09	118.23	114.85	78.20	81.32
0 0 1	115.39	112.62	99.95	99.74	116.86	116.89	101.33	104.56
0 0 2	115.39	112.75	99.95	100.02	116.86	117.02	101.33	104.86
0 0 3	115.19	118.11	99.57	106.82	116.65	122.38	100.95	110.94
0 1 0	82.00	80.38	65.76	68.38	83.43	84.74	67.10	72.76
0 1 1	80.29	79.40	64.13	65.27	81.73	83.76	65.45	69.24
0 1 2	83.09	81.95	71.97	72.04	84.53	86.32	73.40	76.83
0 1 3	78.45	80.85	65.60	69.72	79.89	85.22	66.95	73.82
1 0 0	87.53	84.50	66.05	65.43	88.96	88.86	67.25	69.69
1 0 1	83.31	83.39	72.90	72.76	84.75	87.75	74.26	77.25
1 0 2	82.14	81.97	71.54	71.90	83.57	86.33	72.89	76.61
1 0 3	82.28	86.49	72.48	76.62	83.71	90.86	73.81	80.59
1 1 0	78.98	78.49	62.82	66.05	80.41	82.86	64.12	70.57
1 1 1	76.81	75.97	61.97	63.15	78.25	80.33	63.29	67.24
1 1 2	77.41	76.66	64.57	65.12	78.84	81.03	65.90	69.73
1 1 3	78.82	82.22	66.22	70.68	80.26	86.59	67.55	74.70
2 0 0	80.38	78.38	64.25	65.81	81.82	82.75	65.51	70.78
2 0 1	81.62	81.02	68.23	67.90	83.05	85.38	69.51	72.12
2 0 2	83.15	82.48	69.09	69.27	84.58	86.85	70.39	73.66
2 0 3	79.03	83.55	66.31	70.72	80.46	87.92	67.57	74.67
2 1 0	80.68	80.47	66.31	69.01	82.11	84.84	67.62	73.48
2 1 1	82.17	83.55	70.05	74.29	83.60	87.92	71.40	78.44
2 1 2	81.73	81.51	65.91	65.24	83.16	85.87	67.22	69.66
2 1 3	79.51	83.91	67.26	71.14	80.94	88.28	68.55	75.05
3 0 0	84.69	82.47	64.19	63.45	86.12	86.83	65.38	67.41
3 0 1	79.47	77.95	68.44	68.11	80.91	82.32	69.78	72.88
3 0 2	81.87	80.49	70.92	70.37	83.30	84.85	72.28	75.30
3 0 3	82.97	86.92	72.57	78.62	84.41	91.29	73.94	82.95
3 1 0	78.87	77.09	55.94	55.85	80.30	81.46	57.16	60.17
3 1 1	76.79	75.99	63.63	64.95	78.22	80.35	64.94	69.02
3 1 2	77.39	76.87	66.04	66.30	78.83	81.23	67.37	70.91
3 1 3	77.60	82.37	67.20	72.43	79.03	86.73	68.54	76.53

We observe significant variability. For example, the first branching policy 0-0-0 performs very poorly for all instances, except for *neos-1480121* where it is the best (see Table 10 in the Appendix). We expected poor performance of this policy as

it weights each active clause the same regardless of its size, uses the maximum count of the number of times each variable occurs in these clauses and weights this with the fractionality of the variable. Although there is a clear indication that giving equal weight to each clause (setting $0 - x - x$) is worse than weighting schemes that take clause size into account (settings $1 - x - x$, $2 - x - x$, and $3 - x - x$), there is no clear winner among any of these strategies. We further observe that in general, the policies that simply add the weights of the clauses that a variable occurs in (setting $x - 1 - x$) rather than the policies that use the maximum of the weights of the clauses that a variable occurs in (setting $x - 0 - x$) lead to better results. The changes observed when varying the estimation method for β_j are smaller, and although there is no policy that clearly outperforms the others with respect to the arithmetic means, $x - x - 0$ seems to perform slightly better than the $x - x - 1$, $x - x - 2$ and $x - x - 3$ weighting schemes with respect to the geometric means.

5 Conclusions and further research

We have integrated restart strategies and learning mechanisms for solving binary MILPs by branch-and-bound. Our computational results demonstrate the effectiveness of learning and restart mechanisms in terms of both node counts and solution times. Most of the improved performance can be attributed to branching and propagation and the use of improved clause information. The benefit of collecting more information from an incomplete tree diminishes rapidly with tree size. Clause-informed branching, especially when the importance of the clauses are weighted according to their size, seems to be very beneficial.

Some important topics for further investigation are:

- Is it important to generate minimum cardinality clauses exactly, or should it be done heuristically based on minimal clauses?
- Are there benefits of generating more clauses from each leaf node? (There may be many minimum cardinality clauses corresponding to a leaf node, but we only generate one for each fathomed node. Note that generating multiple clauses will require significant modification in the branching rules as the group of clauses generated from the same leaf node will be highly correlated in terms of the variables involved.)
- Would multiple restarts be beneficial?
- Is it possible to use the clause information to do a partial restart and if so how?
- Should information from open nodes in the incomplete tree be used in the learning phase?

Acknowledgments This research has been supported in part by the National Science Foundation under grant CMMI-0522485 and by the Air Force Office of Scientific Research under grant FA9550-07-1-0177. We thank the referees for their thorough reviews.

Appendix

See Tables 6, 7, 8, 9, 10. See Figs. 5 and 6.

Table 6 Value of the restart strategy with fathoming-based branching

Problem	Collection phase				Improvement phase				Restart phase				Total			
	#	Size	Min	Max	Total	# (%)	Size	Cuts&Prop&Branch		CPLEX		Cuts&Prop&Branch				
								B	I	DS	BB	B	I			
neos-1480121	# Nodes	798	200	38.17	0	0	50 (25.0)	32.63	483513	187946	88346	1711005	484311	188744		
	Time	5.7			0	3.8			1243.9	409.5	394.3	2559.4	1249.6	419		
mas74	# Nodes	1190	200	26.41	0	396	3836	118 (59.0)	17.68	1062473	1130859	1695810	1068579	1135885		
	Time	6.3			0	0.4	15.8		3378.1	3704.5	6319.3	3488.8	3384.4	3726.6		
neos5	# Nodes	1212	200	24.98	0	42	234	144 (72.0)	12.46	693074	1178505	648299	707857	694286		
	Time	8.3			0	0.2	10.1		2116.2	3012.8	2042.4	2223.7	2124.5	3031.2		
markshare_4_0	# Nodes	336	200	18.12	0	102	935	135 (67.5)	12.47	545488	503334	566080	521166	545824		
	Time	0.2			0	0.1	3.5		215.9	199.6	148.1	199.5	216.1	203.3		
neos23	# Nodes	1030	200	21.13	0	10	105	46 (23.0)	17.31	104033	90175	51820	158582	105063		
	Time	27.4			0.1	0.5	29.1		1977.3	1686.6	615.5	2959	2004.7	1743.1		
neos-1228986	# Nodes	1031	200	16.25	0	44	469	106 (53.0)	11.15	35550	31388	117852	137818	36581		
	Time	23.7			0	0.5	22.2		422.9	332.6	1748.2	1458.4	446.6	378.5		
mas76	# Nodes	1195	200	22	0	141	1662	109 (54.5)	14.96	130850	126488	159932	126188	132045		
	Time	5.6			0	0.2	9.3		347.2	314.9	552	312.5	352.8	329.8		
neos-863472	# Nodes	1008	200	26.5	0	41	381	124 (62.0)	13.74	85612	57466	124665	106332	58855		
	Time	33.2			0.1	1.2	80.1		3310.4	2116.1	5532.6	4026.2	3343.6	2229.4		
pk1	# Nodes	1447	200	29.86	0	697	4255	113 (56.5)	19.39	82254	87185	107129	85072	92887		
	Time	7.5			0	0.9	17.5		256.8	276.2	277	257.7	264.3	301.2		
swath2	# Nodes	656	200	46.52	0	1	2	76 (38.0)	28.43	53890	54639	41937	73907	54546		
	Time	43.8			1.2	4.5	465.2		2431.7	2541.6	2228.8	3288.2	2475.5	3050.6		

Table 6 continued

Problem	Collection phase			Improvement phase			# (%)	Size	Restart phase			Total		
	#	Size	Total	Min	Max	Total			Cuts&Prop&Branch			CPLEX		
									B	I	I	DS	BB	B
neos-1211578	#Nodes	419	200	0	42	463	112 (56.0)	9.65	60625	29235	40683	73016	61044	30117
	Time	6.6		0	0.3	18.9			899.3	368.6	575.6	1026.6	905.9	394.1
neos3	#Nodes	538	200	0	51	155	180 (90.0)	7.1	55418	52345	21618	64223	55956	53038
	Time	34.9		0.4	5.5	195.4			3608.7	3432.5	1477.3	4158.3	3643.6	3662.8
neos14	#Nodes	1468	200	0	11	11	2 (1.0)	18.61	61044	52844	70550	61106	62512	54323
	Time	73.6		0	3	85.8			2425.1	2075.3	2154.1	2206.2	2498.7	2234.7
neos-803220	#Nodes	748	200	0	0	0	0 (0.0)	18.34	37720	37720	69387	41066	38468	38468
	Time	27.9		0.1	0.1	14.5			849.8	849.8	1313.9	917.8	877.7	892.2
pg5_34	#Nodes	845	200	0	0	0	0 (0.0)	22.3	39185	39185	41295	37532	40030	40030
	Time	46.3		3.1	3.9	737.4			992.8	992.8	1092.8	1025.2	1039.1	1776.5
neos-1440447	#Nodes	406	200	0	37	943	106 (53.0)	8.98	19849	22359	47358	36721	20255	23708
	Time	13.4		0.1	0.6	33.8			551.3	785.1	4038.2	1082.4	564.7	832.3
neos22	#Nodes	782	200	0	0	0	27 (13.5)	22.38	31632	27596	493	31919	32414	28378
	Time	67.6		2.9	5.5	967.4			1942.3	1724.6	30.4	1898.8	2009.9	2759.6
neos-1173026	#Nodes	428	200	0	21	107	0 (0.0)	9.02	239	239	489	29768	667	774
	Time	14.6		0.1	1.4	63.8			7	7	28.6	2279	21.6	85.4
neos-538867	#Nodes	994	200	0	340	1679	137 (68.5)	17.24	16532	14704	28075	24817	17526	17377
	Time	51.9		0.1	2.3	120.2			531.4	382.5	1018.6	980.3	583.3	554.6
neos-598183	#Nodes	457	200	0	0	0	0 (0.0)	13.53	15708	15708	571	23741	16165	16165
	Time	22.5		0.3	0.7	68.8			601.8	601.8	19.4	817.2	624.3	693.1
stein45	#Nodes	1614	200	0	0	0	0 (0.0)	15.46	19235	19235	21882	18798	20849	20849

Table 6 continued

Problem	Collection phase			Improvement phase			Restart phase			Total				
	#	Size	# (%)	Min	Max	Total	Size	Cuts&Prop&Branch		CPLEX		Cuts&Prop&Branch		
								B	I	DS	BB	B	I	
harp2	Time 16.2	0	0	0	0	3.5	24.65	99.4	10726	99.4	105.8	96.2	115.6	119.1
	# Nodes 588	200	24.65	0	0	0	0 (0.0)	24.65	10726	10726	27245	14144	11314	11314
prod2	Time 31.9	0	0	0	1.7	1.7	8.38	583.6	583.6	583.6	1850.9	650.9	615.5	617.2
	# Nodes 560	200	14.51	0	9	119	126 (63.0)	8.38	14300	13526	17773	13587	14860	14205
prod1	Time 20	0.1	1.1	1.1	54.1	54.1	6.22	342.9	339.4	339.4	396.2	326.9	362.9	413.5
	# Nodes 405	200	15.94	0	3	61	160 (80.0)	6.22	12904	13175	15916	13258	13309	13641
neos-631694	Time 8.2	0.1	0.3	0.1	0.3	22	34.5	198.3	161.7	161.7	165.2	149	206.5	191.9
	# Nodes 821	200	35.24	0	0	0	4 (2.0)	16400	1429	1429	516	11988	17221	2250
neos-860244	Time 117.5	2.3	5.7	5.7	592	592	9.64	1094.2	231.1	231.1	96.3	1639.4	1211.7	940.6
	# Nodes 471	200	29.45	0	9	49	142 (71.0)	9.64	2177	2009	3204	10482	2648	2529
neos-538916	Time 80.8	2.1	6.3	6.3	658.2	658.2	14.83	328	305.5	305.5	435.1	1975.2	408.8	1044.5
	# Nodes 850	200	28.86	0	272	1942	130 (65.0)	14.83	11238	7101	20382	10363	12088	9893
qiu	Time 54	0.1	4.2	4.2	156.7	156.7	11.4	540.6	255.6	255.6	920.1	529.2	594.6	466.3
	# Nodes 1084	200	13.69	0	164	2446	93 (46.5)	11.4	10147	3411	11568	9121	11231	6941
misc07	Time 250.2	0.6	5.3	5.3	425.5	425.5	18.89	1499.5	517.2	517.2	1662.5	1518	1749.7	1192.9
	# Nodes 690	200	31.75	0	198	782	100 (50.0)	18.89	8905	8250	10472	8427	9595	9722
neos-886822	Time 33.4	0.1	1.3	1.3	36.3	36.3	10.11	331.5	264.4	264.4	408.4	305.3	364.9	334.1
	# Nodes 2427	200	25.48	0	3	141	148 (74.0)	10.11	8232	8571	8488	8408	10659	11139
neos-480878	Time 376.4	0.5	5	5	268.1	268.1	13.98	976.7	1043.4	1043.4	1096.4	1002	1353.1	1687.9
	# Nodes 927	200	17.43	0	253	1693	81 (40.5)	13.98	8434	12150	15275	8384	9361	14770
	Time 100.4	0.4	3.4	3.4	149.2	149.2	663	942.9	663	663	1167	657.1	763.4	1192.5

Table 6 continued

Problem	Collection phase				Improvement phase				Restart phase				Total			
	#	Size	# Nodes	Time	Min	Max	Total	# (%)	Size	Cuts& Prop&Branch		CPLX		Cuts& Prop&Branch		
										B	I	DS	BB	B	I	
neos-1200887	591	200	13.78	0	78	1764	129 (64.5)	8.84	4788	2154	10972	7117	5379	4509		
	61.6			0.1	1.6	87.1			329.1	118	731.5	508.1	390.7	266.7		
neos-807639	547	200	13.79	0	0	0	0 (0.0)	13.79	6006	6006	5928	6858	6553	6553		
	37.4			0.1	0.2	34.8			242.6	242.6	189	272.6	280	314.8		
neos2	566	200	64.19	0	28	162	129 (64.5)	31.48	6968	8283	3163	6574	7534	9011		
	36			0.3	5	125.7			330.4	392.5	218.1	316.4	366.4	554.2		
bienst1	609	200	14.99	0	61	1139	140 (70.0)	7.75	5266	2358	5953	5411	5875	4106		
	87.2			0.3	5.2	332.7			546.2	257.9	334.6	521.6	633.4	677.8		
neos-504674	738	200	18.01	0	301	1882	47 (23.5)	15.92	4830	4949	4078	5384	5568	7569		
	83.9			0.5	5.2	311.4			412.8	412.6	546.4	450.9	496.7	807.9		
rout	3086	200	19.8	0	3	34	55 (27.5)	15.62	6185	6337	11115	4865	9271	9457		
	138.8			0.1	1.1	26.8			384.5	366.2	1474.3	322	523.3	531.8		
neos-1396125	597	200	26.98	0	18	232	76 (38.0)	19.62	3904	2144	3474	4660	4501	2973		
	195.1			2.1	5.3	878.1			1140.7	513.2	930.4	1414.7	1335.8	1586.4		
neos21	374	200	16.48	0	56	65	2 (1.0)	16.43	3225	3274	4370	4649	3599	3713		
	83.4			0.2	1.7	61.1			560.8	570.8	803.5	850	644.2	715.3		
neos-892255	716	200	24.3	0	7	132	97 (48.5)	15.5	4419	3989	4340	3642	5135	4837		
	386.5			1	5.4	757.9			1616.3	1416.5	1835.4	1646.6	2002.8	2560.9		
neos-1109824	400	200	21.41	0	94	383	104 (52.0)	14.54	2280	2242	2686	3439	2680	3025		
	56.4			0.4	5.1	303.6			221.6	212.7	582.2	452.8	278	572.7		

Table 6 continued

Problem	Collection phase			Improvement phase				Restart phase				Total			
	#	Size	# Nodes	Min	Max	Total	# (%)	Size	Cuts&Prop&Branch			CPLEX	Cuts&Prop&Branch		
				Time	Time	Time			B	I	DS		BB	B	I
lrn	282	59.17	200	0	0	0	0 (0.0)	59.17	1331	1331	1331	1778	2850	1613	1613
	Time	115.6		5	5.3	1030.2			375.2	375.2	375.2	642.5	642.2	490.8	1521
neos-570431	# Nodes	331	200	0	13	49	6 (3.0)	11.98	887	1137	1137	3579	2433	1218	1517
	Time	74.9		0.2	1.6	66.2			178.9	204.4	204.4	598.9	443.3	253.8	345.5
neos-512201	# Nodes	576	200	0	228	2710	66 (33.0)	13.55	1602	2203	2203	4482	2186	2178	5489
	Time	54.8		0.5	5.2	415.7			117.2	170.4	170.4	596	170.7	172	640.9
neos-806323	# Nodes	639	200	0	0	0	0 (0.0)	14.76	2471	2471	2471	2615	2041	3110	3110
	Time	54.8		0.1	0.2	33.6			191.3	191.3	191.3	186.3	152.7	246.1	279.7
neos-803219	# Nodes	421	200	0	0	0	0 (0.0)	16.95	1893	1893	1893	2591	1987	2314	2314
	Time	17		0.1	0.1	14.5			65.1	65.1	65.1	84.6	68.3	82.1	96.6
neos-504815	# Nodes	522	200	0	517	2306	59 (29.5)	12.65	1389	1412	1412	1381	1575	1911	4240
	Time	36.7		0.3	5.1	317.1			67.5	60.1	60.1	97	75.8	104.2	413.9
neos-820879	# Nodes	421	200	0	7	7	76 (38.0)	15.25	1262	1489	1489	801	1348	1683	1917
	Time	55.9		1.3	5.6	738.2			138.6	152.3	152.3	91.3	147.3	194.5	946.4
neos-807705	# Nodes	649	200	0	0	0	0 (0.0)	11.86	1320	1320	1320	1633	1231	1969	1969
	Time	57.4		0.1	0.2	34.2			107.3	107.3	107.3	125.6	100.6	164.7	198.9
bc1	# Nodes	378	200	0	0	0	44 (22.0)	21.78	608	479	479	1176	1210	986	857
	Time	257		2.4	5.2	806			554	412.7	412.7	981.7	993.7	811	1475.7
seymour1	# Nodes	715	200	0	0	0	0 (0.0)	14.34	1405	1405	1405	1291	1171	2120	2120
	Time	988		5	5.1	1011.9			1584.5	1584.5	1584.5	1399.5	1359.3	2572.5	3584.4

Table 6 continued

Problem	Collection phase			Improvement phase			Restart phase			Total		
	#	Size	# (%)	Min	Max	Total	Size	Cuts&Prop&Branch		CPLEX	Cuts&Prop&Branch	
								B	I		DS	BB
Arith.Mean	#Nodes 795.8	200 23.08	0 84.3	653.6	70.57 (35.3)	16.61	74498.6	76399.6	81421.9	104274.6	75294.4	77849
(All: 51 models)	Time 88	0.7 2.6	249.3	861.4	752.6	1026.7	1116.2	949.4	1089.9			
Geo.Mean	#Nodes 691.2	200 21.07	0 13.7	56.8	24.56 (12.3)	15.14	11688.6	10095.2	11277.7	15729.7	13418.5	12804.8
(All: 51 models)	Time 41.9	0.5 1.9	86.9	494.2	417.2	524.3	677.3	581.3	712.6			
Arith.Mean	#Nodes 948.3	200 26.33	0 121.4	962.1	101.15 (50.6)	16.58	265678.8	275569.9	287286.2	376527	266627.1	277480.3
(Hard: 13 models)	Time 21.3	0.1 1.3	73.6	1741	1574.7	1851.2	2166.5	1762.3	1669.6			
Geo.Mean	#Nodes 866.7	200 24.8	0 35.4	183.9	77.72 (38.9)	15.37	139124.4	119371.1	125247.4	187595.9	140568.2	121816.5
(Hard: 13 models)	Time 12.8	0.1 0.9	27.1	1185.5	965.6	1083.2	1486.4	1203.8	1026			
Arith.Mean	#Nodes 789	200 22.52	0 77.3	648.2	73.87 (36.9)	15.55	13540	12210.6	16108	16689.9	14329	13647.8
(Medium: 23 models)	Time 71.3	0.6 2.8	228.9	608.5	500.5	776.8	866.1	679.8	800.7			
Geo.Mean	#Nodes 710.3	200 20.31	0 12	56.9	21.32 (10.7)	14.13	9465.2	7512.4	7686.5	13503.9	10685.6	9730.4
(Medium: 23 models)	Time 46.3	0.4 2.1	96.4	428.3	342.3	383.3	654.8	498.5	595.5			
Arith.Mean	#Nodes 673.8	200 21.12	0 62.9	394.5	39.00 (19.5)	18.27	2278.7	2208.4	3154.1	2619.1	2952.5	3276.7
(Easy: 15 models)	Time 171.5	1.3 3.5	433	486.9	426.9	695.3	589.3	658.4	1031.4			
Geo.Mean	#Nodes 544.7	200 19.34	0 7.1	20	10.86 (5.4)	16.62	1887.4	1866.3	2519.1	2318.5	2483.5	2768.3
(Easy: 15 models)	Time 97.6	0.8 2.7	200.6	288.1	272.9	451.8	360.8	391.3	684.1			

Table 7 Effect of providing information in different forms: cuts, propagation, branching

Problem	CPLEX		Cuts		Prop		Prop&Branch		Cuts&Prop		Cuts&Prop&Branch	
	DS	BB	B	I	B	I	B	I	B	I	B	I
neos-1480121	# Nodes	88346	1711005	519044	1373113	179692	572460	192307	1408791	165316	483513	187946
	Time	394.3	2559.4	2712.2	1138.3	2160.7	287.3	1207	374.9	2834.9	272.6	1243.9
mas74	# Nodes	1695810	1068579	1068579	1068579	1068566	1062473	1130859	1068566	1068562	1062473	1130859
	Time	6319.3	3488.8	3417.6	3409.8	3638.3	3754.1	3547.6	3862.8	3629	3716.4	3378.1
neos5	# Nodes	648299	707857	707857	707857	677465	693074	1178505	707415	677465	693074	1178505
	Time	2042.4	2223.7	2124.4	2040.9	2222.5	2229.4	2183.4	2975.3	2142.4	2249.8	2116.2
markshare_4_0	# Nodes	566080	521166	521166	506296	521611	526805	50836	521611	506282	545488	503334
	Time	148.1	199.5	199.7	194.3	218.6	228.7	226.1	206.0	213.9	217.3	215.9
neos23	# Nodes	51820	158582	190894	174018	235803	137158	229827	121610	163040	104033	90175
	Time	615.5	2959	3273.1	3518.4	4031.5	2613.6	3436.9	1775.9	2506.7	3402.1	1977.3
neos-1228986	# Nodes	117852	137818	137818	137818	137937	101588	35550	31388	137937	101588	31388
	Time	1748.2	1458.4	1490.9	1442.3	1509.4	1146.7	418.8	323.3	1497.5	1140.4	422.9
mas76	# Nodes	159932	126188	126188	126188	126181	127798	130850	126488	126181	127798	130850
	Time	552.0	312.5	311	319.9	307.6	348.0	337.7	308.5	315.6	355.4	347.2
neos-863472	# Nodes	124665	106332	116146	60422	106715	105666	81350	60753	111297	61668	57466
	Time	5532.6	4026.2	4580.4	2291.3	4029.9	4006.0	3032.1	2134.3	4418.3	2422.7	3310.4
pk1	# Nodes	107129	85072	85072	85072	83694	88063	82254	87185	83694	88063	82254
	Time	277.0	257.7	260.3	258.6	256	278.9	263.6	277.3	255.9	279.6	256.8
swath2	# Nodes	41937	73907	73907	73907	74145	69949	53890	54639	74145	69949	54639
	Time	2228.8	3288.2	3255.8	3286.6	3321.7	3174.4	2437.1	2510.0	3298.7	3157.3	2431.7
neos-1211578	# Nodes	40683	73016	73016	73016	72999	72965	60625	29235	72999	72965	29235
	Time	575.6	1026.6	1010.4	1033.8	1041.6	1030.8	896.9	362.4	1011.2	1045.2	899.3

Table 7 continued

Problem	CPLEX		Cuts		Prop		Prop&Branch		Cuts&Prop		Cuts&Prop&Branch	
	DS	BB	B	I	B	I	B	I	B	I	B	I
neos3	# Nodes	64223	61364	48108	64318	23813	55510	48633	60215	36067	55418	52345
	Time	1477.3	4158.3	4103.5	3294.8	4191	1497.5	3077.1	3964.7	2359.3	3608.7	3432.5
neos14	# Nodes	70550	61106	61043	44868	61054	61013	55164	61120	44834	61044	52844
	Time	2154.1	2206.2	2208.7	1728.8	2208.5	2240.4	2247.8	2224.6	1765.6	2425.1	2075.3
neos-803220	# Nodes	69387	41066	40954	40954	41308	37911	37911	40868	40868	37720	37720
	Time	1313.9	917.8	939.2	939.2	929.5	929.5	844	936.8	936.8	849.8	849.8
pg5_34	# Nodes	41295	37532	39151	39151	39459	39937	39937	38931	38931	39185	39185
	Time	1092.8	1025.2	1200.2	1200.2	996	996.0	1005.5	1191.8	1191.8	992.8	992.8
neos-1440447	# Nodes	47358	36721	33441	24381	36721	20034	26319	33441	26354	19849	22359
	Time	4038.2	1082.4	1126.2	1045.3	1085.9	1403.5	561	1139.3	1197.8	551.3	785.1
neos22	# Nodes	493	31919	31469	33305	33287	32540	25876	32043	31910	31632	27596
	Time	30.4	1898.8	1903.6	2013.5	2029.1	1934.5	1617.4	1971.8	1932.6	1942.3	1724.6
neos-1173026	# Nodes	489	29768	1731	1731	29768	128	128	7174	7174	239	239
	Time	28.6	2279	113.5	113.5	2335.3	2335.3	3.8	604.2	604.2	7	7
neos-538867	# Nodes	28075	24817	24817	24817	24817	22828	14704	24817	22828	16532	14704
	Time	1018.6	980.3	984.5	1017.5	1001.2	920.1	530.4	381.9	1015.1	924.8	382.5
neos-598183	# Nodes	571	23741	18040	18040	23163	17445	17445	17811	17811	15708	15708
	Time	19.4	817.2	600.7	600.7	820.6	820.6	568	590.7	590.7	601.8	601.8
stein45	# Nodes	21882	18798	18798	18798	18781	19235	19235	18781	18781	19235	19235
	Time	105.8	96.2	98.4	98.4	97.3	97.3	97.1	99	99.0	99.4	99.4
harp2	# Nodes	27245	14144	14144	14144	7717	11663	11663	7717	7717	10726	10726
	Time	1850.9	650.9	634.2	634.2	416.8	416.8	633.5	417.4	417.4	583.6	583.6

Table 7 continued

Problem	CPLEX		Cuts		Prop		Prop&Branch		Cuts&Prop		Cuts&Prop&Branch		
	DS	BB	B	I	B	I	B	I	B	I	B	I	
prod2	# Nodes	17773	13587	14183	18117	13873	13327	14015	13560	14068	20600	14300	13526
	Time	396.2	326.9	338.8	454.8	329.6	321.7	335.8	319.1	340	508.1	342.9	339.4
prod1	# Nodes	15916	13258	12647	15240	13080	12532	12989	12923	13060	14708	12904	13175
	Time	165.2	149	198.7	187.9	153.1	143.0	153.1	146.7	199	186.2	198.3	161.7
neos-631694	# Nodes	516	11988	11988	3819	11988	5800	16400	1261	11988	10549	16400	1429
	Time	96.3	1639.4	1629.1	729.9	1633.6	1232.3	1158.9	225.3	1699.1	1686.3	1094.2	231.1
neos-860244	# Nodes	3204	10482	10482	3095	10502	2224	2177	2265	10502	2201	2177	2009
	Time	435.1	1975.2	1996.4	500.0	2013	337.5	327.4	364.1	1973.1	338.7	328	305.5
neos-538916	# Nodes	20382	10363	10363	10363	10363	10231	11238	7101	10363	10231	11238	7101
	Time	920.1	529.2	523.1	524.2	528.8	531.6	540	253.9	531.4	530.1	540.6	255.6
qiu	# Nodes	11568	9121	9121	13979	9114	7091	10147	3447	9114	13605	10147	3411
	Time	1662.5	1518	1413.5	1953.5	1532.4	1238.1	1509.2	563.9	1516.7	1868.0	1499.5	517.2
misc07	# Nodes	10472	8427	8171	6998	9319	8865	7973	8574	8307	9738	8905	8250
	Time	408.4	305.3	318.4	265.0	344.6	331.2	277	270.3	320.6	363.7	331.5	264.4
neos-886822	# Nodes	8488	8408	8306	8340	8340	8268	8411	8686	8301	8143	8232	8571
	Time	1096.4	1002	980.6	1010.2	1019.8	1013.6	1004	1018.6	1011.9	1007.7	976.7	1043.4
neos-480878	# Nodes	15275	8384	8384	9701	8365	7397	8434	13559	8365	9653	8434	12150
	Time	1167.0	657.1	660.8	784.2	648.8	594.6	661.6	1008.9	651.5	796.7	663	942.9
neos-1200887	# Nodes	10972	7117	7117	7117	7117	6572	4788	2154	7117	6572	4788	2154
	Time	731.5	508.1	501.7	522.0	522.7	495.5	335.6	121.3	518.8	488.2	329.1	118
neos-807639	# Nodes	5928	6858	6120	6120	6883	6883	5844	5844	6502	6502	6006	6006
	Time	189.0	272.6	250.4	250.4	277	277.0	233.9	233.9	267.8	267.8	242.6	242.6

Table 7 continued

Problem	CPLEX		Cuts		Prop		Prop&Branch		Cuts&Prop		Cuts&Prop&Branch		
	DS	BB	B	I	B	I	B	I	B	I	B	I	
neos2	# Nodes	3163	6574	6581	5928	6394	5040	7033	8086	6478	5725	6968	8283
	Time	218.1	316.4	320.2	287.9	313.6	244.1	333.7	368.4	314.9	297.5	330.4	392.5
bienst1	# Nodes	5953	5411	5243	2484	5812	3391	5177	2456	5551	2512	5266	2358
	Time	334.6	521.6	512.5	266.9	555.3	321.1	536.2	244.4	560.4	245.7	546.2	257.9
neos-504674	# Nodes	4078	5384	4816	3440	4799	4712	4868	4724	5331	3342	4830	4949
	Time	546.4	450.9	416	307.5	416.6	394.0	403.3	379.6	481.2	306.2	412.8	412.6
rout	# Nodes	11115	4865	3928	3849	15792	6181	5829	5586	4718	3072	6185	6337
	Time	1474.3	322	237.2	257.8	1602.9	433.8	384.7	294.7	316.5	173.0	384.5	366.2
neos-1396125	# Nodes	3474	4660	3724	4387	4627	4376	3175	842	3402	4445	3904	2144
	Time	930.4	1414.7	1098.4	1348.3	1359.1	1269.0	877.6	168.6	983.8	1388.8	1140.7	513.2
neos21	# Nodes	4370	4649	4649	4649	4770	4770	3225	3274	4770	4770	3225	3274
	Time	803.5	850	839.3	814.2	842.9	854.8	573.1	586.1	882.8	869.0	560.8	570.8
neos-892255	# Nodes	4340	3642	3642	3335	3642	3095	4419	3632	3642	3549	4419	3989
	Time	1835.4	1646.6	1630.7	1538.9	1693.3	1446.4	1650.2	1314.9	1621.2	1618.8	1616.3	1416.5
neos-1109824	# Nodes	2686	3439	2126	2048	3439	3969	2327	2449	2126	2147	2280	2242
	Time	582.2	452.8	256.3	230.6	457.2	527.7	225.1	243.0	259.5	245.2	221.6	212.7
lrn	# Nodes	1778	2850	861	861	2431	2431	1580	1580	1199	1199	1331	1331
	Time	642.5	642.2	294.9	294.9	577.1	577.1	431.4	431.4	363	363.0	375.2	375.2
neos-570431	# Nodes	3579	2433	1816	2552	2411	2411	925	2142	1759	2868	887	1137
	Time	598.9	443.3	351.2	505.1	437.1	459.5	180.7	348.8	345.9	582.0	178.9	204.4
neos-512201	# Nodes	4482	2186	1626	1461	1997	2060	1588	1907	1696	1759	1602	2203
	Time	596.0	170.7	124	134.4	163	162.9	113.4	138.1	105.7	169.7	117.2	170.4

Table 7 continued

Problem	CPLEX		Cuts		Prop		Prop&Branch		Cuts&Prop		Cuts&Prop&Branch	
	DS	BB	B	I	B	I	B	I	B	I	B	I
neos-806323	# Nodes	2041	2052	2052	1998	1998	2509	2509	1869	1869	2471	2471
	Time	186.3	152.7	153.9	153.9	148.8	148.8	190.1	190.1	137.2	137.2	191.3
neos-803219	# Nodes	2591	1987	1935	1935	1941	1941	1898	1898	1938	1893	1893
	Time	84.6	68.3	69.5	69.5	66.6	66.6	62.9	62.9	69.5	65.1	65.1
neos-504815	# Nodes	1381	1575	1559	1333	1461	1616	1382	1482	1577	1389	1412
	Time	97.0	75.8	82.9	75.1	71.3	98.1	65.8	60.2	86.2	90.3	60.1
neos-820879	# Nodes	801	1348	1348	993	1144	911	1262	1280	1144	1262	1489
	Time	91.3	147.3	147.5	108.7	124.8	94.3	135.8	121.5	126.2	74.8	152.3
neos-807705	# Nodes	1633	1231	1160	1160	1208	1208	1370	1370	1147	1320	1320
	Time	125.6	100.6	97.3	97.3	99.4	99.4	107.2	107.2	99	107.3	107.3
bc1	# Nodes	1176	1210	874	677	1056	766	757	579	726	608	479
	Time	981.7	993.7	705.5	593.7	942.6	691.3	658.1	486.6	599.7	554	412.7
seymour1	# Nodes	1291	1171	1171	1171	1148	1148	1405	1405	1148	1405	1405
	Time	1399.5	1359.3	1351.2	1351.2	1339.1	1339.1	1587.5	1587.5	1390.3	1584.5	1584.5
Ari th. Mean (All: 51 models)	# Nodes	81421.9	104274.6	98267.5	78190.6	99286	71424.2	78554.3	76915.4	96766.1	74498.6	76399.6
	Time	1026.7	1116.2	1059.8	926.2	1158.1	949.7	877.8	740.1	1059.8	861.4	752.6
Geo. Mean (All: 51 models)	# Nodes	11277.7	15729.7	13826.9	12353.0	15774.6	13243.6	11782.9	10050.7	13971.1	11688.6	10095.2
	Time	524.3	677.3	607.8	554.8	694.4	588.9	488.3	402.8	626.7	560.6	417.2
Ari th. Mean (Hard: 13 models)	# Nodes	287286.2	376527	356392.8	278861.0	356427	249255.9	281874.2	277031.2	350429.3	265678.8	275569.9
	Time	1851.2	2166.5	2226.8	1842.9	2241.3	1756.6	1834.8	1548.5	2177.9	1721.8	1574.7
Geo. Mean (Hard: 13 models)	# Nodes	125247.4	187595.9	188082.9	157273.9	190045	140600.5	149230.1	121213.8	180646.8	139124.4	119371.1
	Time	1083.2	1486.4	1510.9	1295.2	1523.4	1162.6	1229.7	949.6	1495.5	1153.6	965.6

Table 7 continued

Problem	CPLEX		Cuts		Prop		Prop&Branch		Cuts&Prop		Cuts&Prop&Branch	
	DS	BB	B	I	B	I	B	I	B	I	B	I
Arith.Mean	16108.0	16689.9	15046.4	14350.5	16563.9	15801.2	13402.4	12580.3	15070.9	14628.5	13540	12210.6
(Medium: 23 models)	Time	776.8	866.1	767.9	682.9	869.6	753.4	594.4	498.8	797.9	729.8	608.5
Geo.Mean	7686.5	13503.9	11602.5	10202.5	13212	11311.5	9169.7	7460.1	12146.8	10987.0	9465.2	7512.4
(Medium: 23 models)	Time	383.3	654.8	574.9	513.1	651.2	564.5	408.4	332.1	620.9	560.3	342.3
Arith.Mean	3154.1	2619.1	2164.7	2164.2	3271	2592.1	2243.4	2129.0	2190.7	2164.1	2278.7	2208.4
(Easy: 15 models)	Time	695.3	589.3	496	504.9	661.7	551.2	482.9	409.4	492.4	519.7	486.9
Geo.Mean	2519.1	2318.5	1883.1	1825.5	2393.8	2176.1	1916.3	1833.7	1883	1803.3	1887.4	1866.3
(Easy: 15 models)	Time	451.8	360.8	300.4	299.7	387.5	348.3	288.1	257.3	298.8	288.1	272.9

Table 8 Number of times CPLEX branching is overwritten

Problem	# Decisions in		# Decisions in					
	<i>B-Prop</i>	<i>I-Prop</i>	<i>B-Prop&Branch</i>			<i>I-Prop&Branch</i>		
	Prop	Prop	Prop	Branch	%	Prop	Branch	%
neos-1480121	151	5	4	2188	0.38	13	978	0.51
mas74	24	1009	49	578	0.05	2213	52578	4.65
neos5	14	28126	30	1129	0.16	83956	512698	43.50
markshare_4_0	37	218	87	476	0.09	285	8182	1.63
neos23	1488	6582	702	1168	0.51	4632	5244	4.97
neos-1228986	7	4066	15	2010	5.65	996	12462	39.70
mas76	23	545	41	605	0.46	947	15831	12.52
neos-863472	112	6475	116	1034	1.27	1560	6323	10.41
pk1	21	94	50	1067	1.30	345	18547	21.27
swath2	82	27	1	334	0.62	11	329	0.60
neos-1211578	4	1935	19	1142	1.88	981	7825	26.77
neos3	275	47	112	173	0.31	262	1030	2.12
neos14	9	645	24	1018	1.67	19	2416	4.38
neos-803220	105	105	44	387	1.02	44	387	1.02
pg5_34	164	164	189	607	1.52	189	607	1.52
neos-1440447	0	563	16	573	2.86	436	6037	22.94
neos22	50	50	0	1093	4.22	3	1083	3.96
neos-1173026	32	32	3	58	45.31	3	58	45.31
neos-538867	0	569	2	4339	26.25	366	6495	44.17
neos-598183	474	474	57	192	1.10	57	192	1.10
stein45	10	10	137	953	4.95	137	953	4.95
harp2	50	50	71	265	2.27	71	265	2.27
prod2	76	127	71	202	1.44	106	232	1.71
prod1	124	251	78	124	0.95	101	209	1.62
neos-631694	0	1	0	2	0.01	1	2	0.16
neos-860244	25	47	50	211	9.69	39	220	9.71
neos-538916	0	144	18	981	8.73	227	3127	44.04
qiu	23	170	57	567	5.59	59	990	28.72
misc07	12	286	29	718	9.01	380	2947	34.37
neos-886822	10	90	56	1151	13.68	96	1529	17.60
neos-480878	69	607	54	455	5.39	287	2397	17.68
neos-1200887	1	479	61	369	7.71	164	963	44.71
neos-807639	340	340	56	224	3.83	56	224	3.83
neos2	112	117	110	219	3.11	110	543	6.72
bienst1	20	537	77	232	4.48	400	851	34.65
neos-504674	26	337	46	406	8.34	309	1517	32.11

Table 8 continued

Problem	# Decisions in		# Decisions in					
	<i>B-Prop</i>	<i>I-Prop</i>	<i>B-Prop&Branch</i>			<i>I-Prop&Branch</i>		
	Prop	Prop	Prop	Branch	%	Prop	Branch	%
rout	178	292	103	250	4.29	529	1434	25.67
neos-1396125	14	242	18	205	6.46	35	299	35.51
neos21	60	60	179	182	5.64	176	215	6.57
neos-892255	0	91	119	1631	36.91	142	1696	46.70
neos-1109824	0	30	50	390	16.76	87	433	17.68
lrn	69	69	77	76	4.81	77	76	4.81
neos-570431	53	55	156	111	12.00	257	693	32.35
neos-512201	30	273	51	244	15.37	253	822	43.10
neos-806323	21	21	50	303	12.08	50	303	12.08
neos-803219	63	63	77	186	9.80	77	186	9.80
neos-504815	51	164	64	199	14.40	149	489	33.00
neos-820879	16	3	93	170	13.47	77	284	22.19
neos-807705	33	33	55	250	18.25	55	250	18.25
bc1	106	45	97	137	18.10	55	119	20.55
seymour1	17	17	77	318	22.63	77	318	22.63
Arith.Mean(All)	91.78	1113.37	74.47	625.53	0.80	1999.16	13409.57	17.43
Geo.Mean(All)	26.94	151.36	41.67	375.12	3.18	140.90	966.61	9.62
Arith.Mean(Hard)	172.85	3828.77	96.15	994.00	0.35	7401.54	49572.54	17.89
Geo.Mean(Hard)	42.09	561.49	35.68	816.67	0.55	494.70	7304.12	6.03
Arith.Mean(Medium)	74.91	241.30	55.74	622.96	4.65	158.30	1383.83	11.00
Geo.Mean(Medium)	21.99	134.48	31.06	332.80	3.63	83.51	586.21	7.86
Arith.Mean(Easy)	47.40	97.20	84.40	310.13	13.82	139.73	507.80	23.85
Geo.Mean(Easy)	24.88	57.81	74.38	229.42	11.97	105.22	359.92	19.63

Table 9 Results for collecting and using more fathoming information

	Clauses collected		Clauses improved		Collection phase		Improvement phase		Restart phase		
	#	Size	#	Size	# Nodes	Time	# Nodes	Time	# Nodes	Time	
markshare_4_0	<i>CPLEX-BB</i>									521166	199.52
	<i>CPLEX-DS</i>									566080	148.06
	10	13.20	8	10.00	49	0.03	1	0.12	555338	218.63	
	100	17.78	75	11.68	186	0.07	405	1.80	490470	198.28	
	200	18.12	135	12.47	336	0.16	935	3.53	503334	199.56	
	500	18.83	343	12.72	716	0.32	2534	8.95	495809	200.09	
1000	19.52	660	13.28	1369	0.60	5020	17.89	501697	219.34		
mas74	<i>CPLEX-BB</i>									1068579	3488.83

Table 9 continued

	Clauses collected		Clauses improved		Collection phase		Improvement phase		Restart phase	
	#	Size	#	Size	# Nodes	Time	# Nodes	Time	# Nodes	Time
	<i>CPLEX-DS</i>								1695810	6319.29
	10	27.60	8	14.50	102	0.45	317	1.02	989581	3010.77
	100	25.69	52	18.51	615	2.67	2087	7.37	1138897	3542.71
	200	26.41	118	17.69	1190	6.27	3836	15.77	1130859	3704.51
	500	27.11	294	18.08	2737	22.81	11778	41.66	1205746	4079.91
	1000	27.13	612	17.67	5245	94.72	28725	89.03	1194174	4352.65
mas76	<i>CPLEX-BB</i>								126188	312.48
	<i>CPLEX-DS</i>								159932	551.98
	10	22.90	8	11.90	95	0.44	119	0.60	130336	317.31
	100	23.00	52	16.02	647	2.95	1081	4.76	131033	319.96
	200	22.00	109	14.96	1195	5.64	1662	9.27	126488	314.93
	500	21.61	292	13.97	2621	17.68	3749	22.70	157512	411.20
	1000	21.79	573	14.17	4612	52.97	7001	45.46	155997	448.78
neos-1211578	<i>CPLEX-BB</i>								73016	1026.63
	<i>CPLEX-DS</i>								40683	575.56
	10	10.00	6	7.60	39	2.61	5	2.01	47855	1818.27
	100	12.98	53	9.55	230	10.59	183	24.36	34838	1250.06
	200	13.69	112	9.66	419	6.55	463	18.93	29235	368.56
	500	14.30	266	10.17	951	38.91	1007	120.88	40298	1404.29
	1000	14.76	539	10.32	1976	79.16	1673	241.19	35356	1215.76
neos-1228986	<i>CPLEX-BB</i>								137818	1458.37
	<i>CPLEX-DS</i>								117852	1748.15
	10	10.90	7	6.70	50	1.41	6	1.54	52175	582.33
	100	14.99	56	9.99	555	13.08	225	12.05	29737	306.52
	200	16.25	106	11.16	1031	23.66	469	22.18	31388	332.64
	500	17.12	256	11.99	2519	56.48	959	54.00	18826	314.22
	1000	17.49	542	11.96	4242	96.10	2031	113.41	15581	204.04
neos-1480121	<i>CPLEX-BB</i>								1711005	2559.37
	<i>CPLEX-DS</i>								88346	394.29
	10	33.60	4	23.20	92	1.05	0	0.16	413084	1057.73
	100	29.32	47	19.05	408	3.18	0	1.86	292622	904.54
	200	38.17	50	32.63	798	5.67	0	3.85	187946	409.54
	500	45.62	72	41.54	1685	10.11	0	9.81	244621	837.22
	1000	44.35	164	39.10	3072	18.81	0	19.59	251375	1006.89
neos14	<i>CPLEX-BB</i>								61106	2206.18
	<i>CPLEX-DS</i>								70550	2154.09
	10	17.20	1	15.90	119	6.03	0	6.03	52066	1894.66
	100	18.45	1	18.32	765	37.03	0	42.66	53045	1998.21

Table 9 continued

	Clauses collected		Clauses improved		Collection phase		Improvement phase		Restart phase		
	#	Size	#	Size	# Nodes	Time	# Nodes	Time	# Nodes	Time	
neos23	200	18.73	2	18.61	1468	73.57	11	85.79	52844	2075.28	
	500	18.70	4	18.60	3158	160.21	16	211.93	55624	2243.42	
	1000	18.45	16	18.28	5572	314.07	83	431.67	57515	2445.37	
	<i>CPLEX-BB</i>								158582	2959.02	
	<i>CPLEX-DS</i>								51820	615.54	
	10	14.40	4	11.00	107	3.11	5	1.76	52851	948.91	
	100	19.34	25	16.10	586	16.49	58	14.89	93090	1390.19	
	200	21.13	46	17.31	1030	27.43	105	29.15	90175	1686.58	
	500	21.27	95	18.20	2187	61.40	221	70.22	44058	885.46	
	1000	22.20	180	19.10	3632	99.60	453	138.38	58041	1430.29	
neos3	<i>CPLEX-BB</i>								64223	4158.27	
	<i>CPLEX-DS</i>								21618	1477.31	
	10	97.50	7	30.50	337	24.10	35	22.49	64497	4383.01	
	100	40.38	84	10.64	433	30.04	155	117.74	65631	4471.54	
	200	39.98	180	7.10	538	34.91	155	195.42	52345	3432.49	
	500	55.72	405	15.92	1396	85.09	244	481.69	55079	3705.66	
	1000	69.98	834	18.08	2468	148.69	491	936.04	59075	3974.79	
	<i>CPLEX-BB</i>								707857	2223.66	
	<i>CPLEX-DS</i>								648299	2042.44	
	neos5	10	25.40	9	8.20	63	0.50	6	0.46	522599	1706.32
100		25.55	76	12.10	616	4.38	147	5.26	1057156	2772.46	
200		24.98	144	12.47	1212	8.32	234	10.08	1178505	3012.84	
500		25.26	344	13.08	2803	25.62	573	24.26	1092201	3392.59	
1000		26.91	676	13.89	4608	62.23	1524	49.28	876418	4550.92	
neos-863472		<i>CPLEX-BB</i>								106332	4026.16
		<i>CPLEX-DS</i>								124665	5532.57
		10	14.30	6	7.10	47	1.67	8	4.69	101335	3592.25
		100	21.68	67	11.04	498	14.47	122	40.23	59535	2190.73
		200	26.50	124	13.75	1008	33.22	381	80.08	57466	2116.06
	500	30.84	308	15.59	2155	76.92	789	188.74	56136	2115.58	
	1000	31.79	595	16.50	4082	162.16	2411	384.88	48185	2084.20	
	<i>CPLEX-BB</i>								85072	257.67	
	<i>CPLEX-DS</i>								107129	276.97	
	pk1	10	16.20	7	9.90	82	0.40	14	0.57	75300	230.45
100		27.47	58	17.65	641	2.79	1854	7.90	81433	263.40	
200		29.86	113	19.39	1447	7.48	4255	17.51	87185	276.23	
500		32.64	287	20.73	2908	22.31	19525	53.72	87214	271.47	

Table 9 continued

	Clauses collected		Clauses improved		Collection phase		Improvement phase		Restart phase	
	#	Size	#	Size	# Nodes	Time	# Nodes	Time	# Nodes	Time
swath2	1000	32.09	579	20.36	4843	63.26	31542	100.74	85137	294.40
	<i>CPLEX-BB</i>								73907	3288.24
	<i>CPLEX-DS</i>								41937	2228.85
	10	36.80	5	23.60	91	7.26	2	28.46	57699	2608.63
	100	46.93	39	28.48	410	30.75	2	244.54	73333	3301.31
	200	46.53	76	28.43	656	43.80	2	465.20	54639	2541.62
	500	46.26	201	27.92	1186	68.37	2	1174.08	55826	2561.50
	1000	47.36	352	31.00	2320	125.68	2	2331.50	55009	2500.33

Table 10 Comparison of different branching rules

Instance	neos-1480121		markshare_4_0		mas74		mas76		neos-1211578		neos-1228986		neos14	
	# Nodes	Time	# Nodes	Time	# Nodes	Time	# Nodes	Time	# Nodes	Time	# Nodes	Time	# Nodes	Time
<i>CPLEX-BB</i>	1711005	2559.4	521166	199.5	1068579	3488.8	126188	312.5	73016	1026.6	137818	1458.4	61106	2206.2
<i>CPLEX-DS</i>	88346	394.3	566080	148.1	1695810	6319.3	159932	552.0	40683	575.6	117852	1748.2	70550	2154.1
0 0 0	17534	36.5	549777	236.3	2565355	8331.7	237325	528.3	51965	574.9	138314	1320.8	56038	1615.7
0 0 1	758611	1337.8	519487	216.4	2206070	7242.9	225329	523.3	45861	512.4	86486	801.3	73817	2939.5
0 0 2	758611	1340.4	519487	221.9	2206070	7249.3	225329	511.7	45861	520.3	86486	806.7	73817	2943.8
0 0 3	758611	1351.9	519487	215.3	2206070	7066.6	225329	507.8	45861	526.5	86486	833.2	73817	3046.6
0 1 0	248686	740.9	515253	215.8	1234792	3984.9	136498	319.0	27953	365.3	33854	344.3	52879	1933.3
0 1 1	181807	377.6	506762	209.1	1109522	3625.8	131002	315.0	26619	335.4	30615	327.9	54423	2163.3
0 1 2	796786	1204.9	523997	219.6	1117438	3693.0	129931	308.9	29639	369.9	33766	348.7	57964	2090.7
0 1 3	439069	645.3	522452	211.3	1142533	3802.2	134446	316.1	33130	389.4	34871	363.4	57150	2297.7
1 0 0	43965	79.8	551287	242.1	1293548	4285.5	156522	362.8	31955	396.1	81830	775.8	55628	1775.0
1 0 1	468725	707.4	516363	218.7	1161677	3870.4	144213	345.7	25718	303.0	45084	423.4	58414	2175.2
1 0 2	478332	798.8	527078	228.0	1304613	4292.5	141494	349.5	29492	331.2	44709	427.7	59058	2249.6
1 0 3	459352	718.8	518931	212.5	1182208	3886.7	140405	336.6	28428	336.8	46716	437.7	58407	2247.8
1 1 0	189542	543.1	525383	223.8	1206303	4111.9	137729	329.9	28055	364.8	38105	395.8	52150	1865.7
1 1 1	188823	403.2	504477	208.4	1118096	3708.4	130100	319.6	28678	359.7	27275	289.5	53406	2079.8
1 1 2	357131	519.1	515125	222.6	1149457	3795.1	132473	313.0	27293	344.3	38737	398.6	57549	2091.8
1 1 3	327387	503.7	522256	217.2	1132087	3621.1	131181	312.1	31643	370.3	35435	362.9	56516	2240.3
2 0 0	154375	416.1	522047	225.1	1230660	3760.9	151139	365.8	23615	298.3	65629	597.2	54012	1794.1
2 0 1	221596	331.7	548933	234.7	1141976	3658.3	155819	372.4	26007	324.4	46362	443.7	58130	2164.1
2 0 2	216935	319.8	499564	210.3	1241280	4045.8	136762	327.1	27499	382.6	41390	402.6	58420	2189.5
2 0 3	225994	347.0	514733	214.7	1220537	4095.1	139539	335.4	28383	339.8	47208	440.5	58401	2310.4

Table 10 continued

Instance	neos-1480121		markshare_4_0		mas74		mas76		neos-1211578		neos-1228986		neos14	
	# Nodes	Time	# Nodes	Time	# Nodes	Time	# Nodes	Time	# Nodes	Time	# Nodes	Time	# Nodes	Time
2 1 0	206275	586.1	529317	223.4	1216750	4492.0	146341	351.3	21177	274.1	48599	458.1	51416	1785.5
2 1 1	308781	971.9	532446	224.9	1151198	3714.1	133330	331.5	26195	332.9	38281	375.1	52720	2018.1
2 1 2	165427	216.4	510626	212.5	1193989	3861.2	130925	323.2	26172	327.1	34553	334.2	56440	2062.4
2 1 3	264460	391.0	524261	212.2	1183494	4093.2	130229	305.2	31308	354.6	43752	418.7	56201	2156.3
3 0 0	51596	82.9	492079	212.5	1262014	4190.0	150202	378.5	28183	360.7	74634	715.0	55486	1756.7
3 0 1	489340	740.6	501219	206.5	1213767	3978.6	137179	335.0	29754	364.2	50798	504.7	59344	2151.0
3 0 2	551640	873.1	510224	217.0	1263149	4161.6	130904	323.2	29349	333.7	51609	509.2	59334	2164.9
3 0 3	551640	915.7	509511	209.5	1245883	4134.6	129198	314.3	30859	354.3	55410	543.3	59341	2264.8
3 1 0	23063	41.2	525676	221.5	1184422	4049.5	136781	318.3	26333	337.4	45103	459.6	52454	1813.0
3 1 1	187946	409.5	503334	199.6	1130859	3704.5	126488	314.9	29235	368.6	31388	332.6	52844	2075.3
3 1 2	333776	516.6	517149	222.3	1120143	3724.5	123465	303.0	28277	354.1	38661	385.0	56985	2082.0
3 1 3	407426	686.8	515805	213.6	1151893	3842.0	126103	304.7	28881	341.0	36829	361.4	55574	2105.3
Instance	neos23		neos3		neos5		neos-863472		pk1		swath2			
	# Nodes	Time	# Nodes	Time	# Nodes	Time	# Nodes	Time	# Nodes	Time	# Nodes	Time	# Nodes	Time
<i>Cplex-BB</i>	158582	2959.0	64223	4158.3	707857	2223.7	106332	4026.2	85072	257.7	73907	3288.2		
<i>Cplex-DS</i>	51820	615.5	21618	1477.3	648299	2042.4	124665	5532.6	107129	277.0	41937	2228.9		
0 0 0	100393	2075.7	68867	5057.0	2094950	5315.1	78064	2988.3	106348	316.2	41062	2004.9		
0 0 1	144538	3337.8	—	—	1801991	4290.8	79310	3032.0	113699	351.2	40924	1979.7		
0 0 2	144538	3334.3	—	—	1801991	4321.7	79310	3052.0	113699	349.3	40924	1976.3		
0 0 3	144538	3617.1	—	—	1801991	4231.6	79310	3095.8	113699	353.5	39095	3895.6		

Table 10 continued

Instance	neos23		neos3		neos5		neos-863472		pk1		swath2	
	# Nodes	Time	# Nodes	Time	# Nodes	Time	# Nodes	Time	# Nodes	Time	# Nodes	Time
0 1 0	66250	1172.5	52228	3631.3	1657584	4154.6	61907	2308.1	86314	272.6	46230	2227.5
0 1 1	69354	1205.2	48261	3323.6	1434448	3758.5	60533	2246.7	90266	286.5	70944	3222.7
0 1 2	90957	1824.3	59768	4084.6	1459104	3773.8	59181	2198.9	93582	297.6	32066	1614.0
0 1 3	61177	1336.0	49094	3461.9	1427416	3616.9	59242	2177.1	90437	289.3	27807	2809.7
1 0 0	114353	2107.4	55795	3854.7	1462784	3669.4	64580	2393.3	92472	289.2	40802	2045.0
1 0 1	118877	2640.9	76399	5200.1	1147562	2914.9	60246	2235.7	90910	283.8	37341	1875.2
1 0 2	105646	2353.9	48916	3352.2	1271969	3208.7	65818	2379.9	91321	283.2	32018	1585.5
1 0 3	113044	2560.1	58980	4142.5	1198360	2915.3	63530	2326.3	94049	297.0	37052	3803.1
1 1 0	67880	1270.2	48352	3326.2	1543945	4009.9	60492	2301.3	86294	271.5	34636	1662.9
1 1 1	65717	1134.1	49699	3421.6	1300260	3257.8	58635	2184.7	87743	280.7	55967	2608.3
1 1 2	70084	1575.6	48029	3230.7	1391487	3604.2	59927	2266.8	88310	276.0	28592	1437.7
1 1 3	81330	1899.5	48316	3484.7	1393404	3525.0	62879	2348.9	90097	287.6	33624	3213.7
2 0 0	103872	1996.4	59603	4222.0	1257659	3160.0	60080	2208.4	95945	300.5	20127	1024.2
2 0 1	96540	2024.6	72436	5005.7	1119178	2775.1	59102	2178.6	92984	291.1	37936	1900.7
2 0 2	119607	2550.0	73675	4934.7	1247392	3098.4	67868	2534.6	90955	283.5	35093	1725.3
2 0 3	108035	2724.5	43111	3061.7	1227191	3053.9	59621	2140.8	94299	295.4	34269	3335.6
2 1 0	113932	2120.8	50609	3511.5	1345051	3388.2	59797	2175.2	90719	284.6	39466	1909.6
2 1 1	89430	1980.7	64203	4359.3	1208661	3058.6	58637	2148.4	90906	288.2	70604	3248.6
2 1 2	129408	2835.4	76478	5091.3	1241326	3231.1	60324	2264.4	90038	284.1	35080	1716.8
2 1 3	98389	2287.5	44188	3158.1	1348723	3460.1	60573	2228.5	90082	285.0	39022	3894.0

Table 10 continued

Instance	neos23		neos3		neos5		neos-863472		pk1		swath2	
	# Nodes	Time	# Nodes	Time	# Nodes	Time	# Nodes	Time	# Nodes	Time	# Nodes	Time
3 0 0	73051	1445.1	55893	3877.1	1422365	3613.7	60850	2230.3	93439	286.8	60376	2828.4
3 0 1	67214	1403.7	66867	4546.3	1203288	2994.0	59899	2162.8	90157	281.2	24075	1245.6
3 0 2	85956	1709.3	66551	4435.5	1276960	3250.8	59523	2151.0	94446	293.9	23824	1220.4
3 0 3	115810	2637.2	67469	4661.4	1256291	3316.2	63011	2535.9	90315	287.4	21519	2329.0
3 1 0	88218	1424.1	46198	3306.4	1360340	3492.3	65938	2554.0	87351	273.7	47897	2327.1
3 1 1	90175	1686.6	52345	3432.5	1178505	3012.8	57466	2116.1	87185	276.2	54639	2541.6
3 1 2	101024	1975.8	54658	3707.6	1237898	3171.4	64995	2458.9	86295	271.3	28411	1439.9
3 1 3	92562	2085.5	51184	3584.2	1211536	3223.6	65163	2601.5	89369	284.2	33643	3266.4

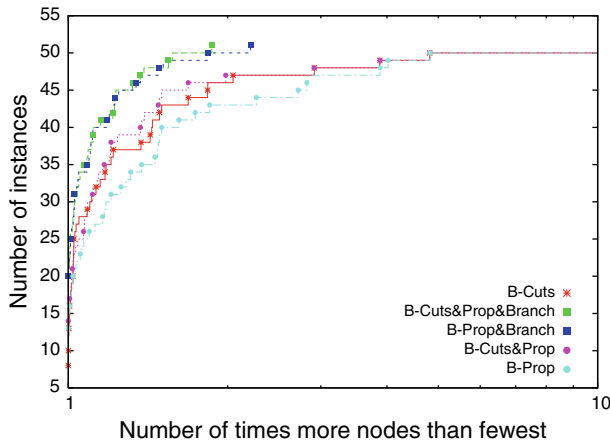


Fig. 5 Comparison of # of nodes in the restart phase for solvers with basic information

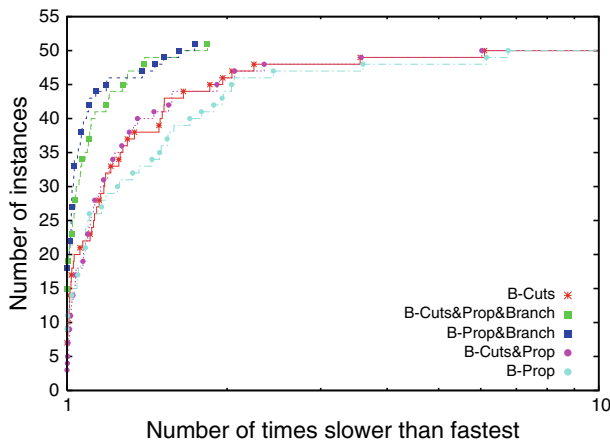


Fig. 6 Comparison of solution times in the restart phase for solvers with basic information

References

1. Achterberg, T.: Conflict analysis in mixed integer programming. *Discrete. Optim.* **4**, 4–20 (2007a)
2. Achterberg, T.: Constraint integer programming. Ph.D. thesis, Technische Universität Berlin. <http://opus.kobv.de/tuberlin/volltexte/2007/1611/> (2007b)
3. Achterberg, T.: SCIP: solving constraint integer programs. *Math. Program. Comput.* **1**, 1–32 (2009)
4. Achterberg, T., Berthold, T.: Hybrid branching. In: van Hoes, W.J., Hooker, J.N. (eds.) CPAIOR, *Lecture Notes in Computer Science*, vol. 5547, pp. 309–311. Springer, Berlin (2009)
5. Achterberg, T., Koch, T., Martin, A.: Branching rules revisited. *Oper. Res. Lett.* **33**, 42–54 (2005)
6. Achterberg, T., Koch, T., Martin, A.: Miplib 2003. *Oper. Res. Lett.* **34**, 361–372 (2006)
7. Amaldi, E., Pfetsch, M.E., Trotter, L.E.: On the maximum feasible subsystem problem, IISs, and IIS-hypergraphs. *Math. Program.* **95**, 533–554 (2003)
8. Applegate, D., Bixby, R., Chvatal, V., Cook, W.: On the solution of traveling salesman problems. *Documenta Mathematica Journal der Deutschen Mathematiker-Vereinigung*. International Congress of Mathematicians, 645–656 (1998)

9. Avenali, A.: Resolution branch and bound and an application: the maximum weighted stable set problem. *Oper. Res.* **55**, 932–948 (2007)
10. Beale, E.M.L., Tomlin, J.A.: Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables. In: *Proceedings of the Fifth International Conference on Operational Research*, vol. 55, pp. 447–454. Tavistock publication, London, UK (1970)
11. Bénichou, M., Gauthier, J.M., Girodet, P., Hentges, G., Ribière, G., Vincen, O.: Experiments in mixed-integer linear programming. *Math. Program.* **1**, 76–94 (1971)
12. Bixby, R.E., Ceria, S., McZeal, C.M., Savelsbergh, M.W.P.: An updated mixed integer programming library: MIPLIB 3.0. *Optima* **58**, 12–15 (1998)
13. Chvátal, V.: Resolution search. *Discrete Appl. Math.* **73**, 81–99 (1997)
14. Cor@l: Computational optimization research at lehigh university. <http://coral.ie.lehigh.edu/mip-instances/> (2009)
15. Cornuéjols, G., Liberti, L., Nannicini, G.: Improved strategies for branching on general disjunction. *Optimization online*, http://www.optimization-online.org/DB_FILE/2008/08/2071.pdf (2008)
16. CPLEX. 11.1. ILOG. <http://www.ilog.com/products/cplex>
17. Davey, B., Boland, N., Stuckey, P.J.: Efficient intelligent backtracking using linear programming. *INFORMS J. Comput.* **14**, 373–386 (2002)
18. Dixon, H.E., Ginsberg, M.L.: Combining satisfiability techniques from AI and OR. *Knowl. Eng. Rev.* **15**, 31–45 (2000)
19. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. *Math. Program.* **91**, 201–213 (2002)
20. Forrest, J.J.H., Hirst, J.P.H., Tomlin, J.A.: Practical solution of large scale mixed integer programming problems with UMPIRE. *Manag. Sci.* **20**, 736–773 (1974)
21. Gilpin, A., Sandholm, T.: Information-theoretic approaches to branching in search. In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. Hyderabad, India (2007)
22. Glankwamdee, W., Linderth, J.T.: Lookahead branching for mixed integer programming. *Optimization online*, http://www.optimization-online.org/DB_FILE/2006/10/1490.pdf (2006)
23. Gleeson, J., Ryan, J.: Identifying minimally infeasible subsystem of inequalities. *ORSA J. Comput.* **2**, 61–63 (1990)
24. Gomes, C., Kautz, H., Sabharwal, A., Selman, B.: Satisfiability solvers. In: *Handbook of Knowledge Representation, Foundations of Artificial Intelligence*, vol. 3, pp. 89–134. Elsevier, Amsterdam (2008)
25. Hooker, J.N.: Constraint satisfaction methods for generating valid cuts. In: Woodruff, D.L. (ed.) *Advances in Computational and Stochastic Optimization, Logic Programming, and Heuristic Search: Interfaces in Computer Science and Operations Research*, pp. 1–30. Kluwer, Norwell (1998)
26. Hooker, J.N.: *Logic-Based Methods for Optimization: Combining Optimization and Constraint Satisfaction*. Wiley, New York (2000)
27. Karamanov, M., Cornuéjols, G.: Branching on general disjunctions. Technical report, <http://integer.tepper.cmu.edu/webpub/d-branching.pdf> (2005)
28. Kiliç Karzan, F., Nemhauser, G.L., Savelsbergh, M.W.P.: Information based branching rules in integer programming. *INFORMS Annual Meeting*. Washington, DC, USA (2008)
29. Linderth, J.T., Savelsbergh, M.W.P.: A computational study of search strategies for mixed integer programming. *INFORMS J. Comput.* **11**, 173–187 (1999)
30. Mahajan, A., Ralphs, T.K.: Experiments with branching using general disjunctions. *Optimization online*, http://www.optimization-online.org/DB_FILE/2008/06/2020.pdf (2008a)
31. Mahajan, A., Ralphs, T.K.: On the complexity of branching on general hyperplanes for integer programming. *Optimization online*, http://www.optimization-online.org/DB_FILE/2008/10/2119.pdf (2008b)
32. Marques-Silva, J.P., Sakallah, K.A.: Grasp: a search algorithm for propositional satisfiability. *IEEE Trans. Comput.* **48**, 506–521 (1999)
33. MINTO. 3.1. MINTO: Mixed INTEger Optimizer. Georgia Institute of Technology, <http://coral.ie.lehigh.edu/minto/>
34. Mittelman, H.D.: Milptestset. <http://plato.asu.edu/ftp/milp/> (2009)
35. Moskewicz, M.W., Madigan, C.F., Zhao, Y., Zhang, L., Malik, S.: Chaff: engineering an efficient sat solver. In: *DAC '01: Proceedings of the 38th Annual Design Automation Conference* (2001)
36. Owen, J.H., Mehrotra, S.: Experimental results on using general disjunctions in branch-and-bound for general-integer linear programs. *Comput. Optim. Appl.* **20**, 159–170 (2001)
37. Patel, J., Chinneck, J.W.: Active-constraint variable ordering for faster feasibility of mixed integer linear programs. *Math. Program.* **110**, 445–474 (2007)

38. Sandholm, T., Shields, R.: Nogood learning for mixed integer programming. Technical report, CMU-CS-06-155, Carnegie Mellon University, Computer Science Department, <http://www.cs.cmu.edu/sandholm/nogoodsForMip.techReport06.pdf> (2006)
39. SCIP. 1.1.0. SCIP: solving constraint integer programs. Zuse Institute Berlin, <http://scip.zib.de>
40. Stallman, R.M., Sussman, G.J.: Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis. *Artif. Intell.* **9**, 135–196 (1977)