Experiments with Branching using General Disjunctions

A. Mahajan and T.K. Ralphs

Abstract Branching is an important component of the branch-and-cut algorithm for solving mixed integer linear programs. Most solvers branch by imposing a disjunction of the form"$x_i \leq k \vee x_i \geq k+1$" for some integer k and some integer-constrained variable x_i. A generalization of this branching scheme is to branch by imposing a more general disjunction of the form "$\pi x \leq \pi_0 \vee \pi x \geq \pi_0 + 1$." In this paper, we discuss the formulation of two optimization models for selecting such a branching disjunction and then describe methods of solution using a standard MILP solver. We report on computational experiments carried out to study the effects of branching on such disjunctions.

Key words: branching, integer programming, general disjunctions

1 Introduction

In this paper, we consider the effect of using more general branching disjunctions in the well-known branch-and-cut algorithm for solving mixed integer linear programs (MILPs) than are typically considered by most solvers. Even though the method of selecting a branching disjunction is a crucial component of branch and cut, most solvers still only consider a very limited set of possible disjunctions when deciding how to branch. It is not clear whether the reason for this is (i) that it is not known how to generate more general

A. Mahajan
Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA 18015, e-mail: asm4@lehigh.edu

T.K. Ralphs
Department of Industrial and Systems Engineering Lehigh University, Bethlehem, PA 18015, e-mail: ted@lehigh.edu

branching disjunctions or (ii) the additional effort necessary to generate such disjunctions is not offset by gains in the overall efficiency of the algorithm. In what follows, we address this question by formally stating the problem of selecting a "best" branching disjunction as an optimization problem, proposing a method to solve this optimization problem, and reporting on the effect of deploying this method of selection in a standard commercial solver. Our goal here is not to test the efficiency of our method for selecting disjunctions (it is demonstrably inefficient), but simply to answer the question of what gains could be realized in the overall efficiency of a branch-and-cut procedure (in terms of reducing the number of subproblems solved during the solution procedure) if "optimal" branching disjunctions could be determined.

1.1 Definitions

We consider the mixed integer linear program

$$\begin{aligned} &\min\ cx \\ &\text{s.t. } Ax \geq b \\ &\qquad x \in \mathbb{Z}^d \times \mathbb{R}^{n-d}, \end{aligned} \tag{P}$$

where $b \in \mathbb{Q}^m, c \in \mathbb{Q}^n$, and $A \in \mathbb{Q}^{m \times n}$ are the inputs and the variables with indices $1, 2, \ldots, d$ are required to take on integral values. If (P) does not have any feasible solution, then the optimal solution value is taken to be ∞. The linear programming (LP) relaxation of (P), obtained by dropping the integrality constraints, is the linear program

$$\min_{x \in \mathcal{P}} cx, \tag{P^{LP}}$$

where $\mathcal{P} = \{x \in \mathbb{R}^n \mid Ax \geq b\}$. Formally defining the problem of determining the "best" branching disjunction requires defining precisely what set of possible disjunctions we consider and by what criteria we evaluate them. To do this, we must first briefly describe the branch-and-bound procedure.

LP-based branch and bound is a recursive procedure for solving (P) in which a lower bound is first obtained by solving its LP relaxation (P^{LP}) (with the minimum taken to be ∞ if $\mathcal{P}$ is empty). If the bound obtained is at least as large as the value of the best feasible solution known (generated either by a separate heuristic procedure or as a by-product of solving the relaxation), then the current best solution is globally optimal and we are done. Otherwise, we determine a disjunction (usually binary) that is satisfied by all solutions to the original MILP, but not satisfied by the solution to the LP relaxation. Such a disjunction, referred henceforth to as a *valid branching disjunction*, is then used to partition the feasible region into subsets that define subproblems to which the algorithm can then be applied recursively until exhaustion. For

a more complete description of the algorithm (and also of the branch-and-cut algorithm), see [Nemhauser and Wolsey, 1988, page 355]. Note that a *subproblem* refers to a restriction of the original problem resulting from the imposition of one or more branching disjunctions on the original instance. These subproblems should not be confused with the associated problem of selecting a branching disjunction, which is formulated below and then solved to determine an optimal branching disjunction.

Most solvers use branching disjunctions, called *variable disjunctions*, of the form "$x_i \leq k \vee x_i \geq k+1$" for some integer k and some $i \leq d$, since these are always valid for (P). More generally, however, any $\pi \in \mathbb{Z}^d \times \{0\}^{n-d}$ and $\pi_0 \in \mathbb{Z}$ yields the disjunction "$\pi x \leq \pi_0 \vee \pi x \geq \pi_0 + 1$" (referred henceforth to as a *general disjunction* and denoted by the ordered pair (π, π_0)), which is also always valid for (P). Since the set of general disjunctions includes all variable disjunctions, considering this larger set should in principle be advantageous. It is this set of disjunctions we consider in what follows, though in the computational experiments, we were forced to further restrict the set in order to obtain results in a reasonable amount of time.

In its simplest form, the efficiency of the branch-and-bound procedure depends mainly on the number of subproblems generated. The goal of selecting the branching disjunctions is then to minimize the total number of subproblems to be solved. It is evident that the problem of selecting a branching disjunction that minimizes the total number of subproblems solved globally is extremely difficult—at least as hard as solving the original problem and likely much harder in practice. The approach taken by most solvers, and the one we shall take here, is to evaluate candidate branching disjunctions by assessing their effect using more myopic criteria. We defer discussion of the specific criteria employed in this study until Section 2 below.

1.2 Previous Work

Despite its importance as a component of the branch-and-bound procedure, relatively little effort has gone into improving methods by which branching disjunctions are determined. In practice, however, where branching is typically limited to variable disjunctions, some attention has been paid to selecting the "best" such disjunction. Linderoth and Savelsbergh [1999] performed extensive computational experiments to show that selecting a variable disjunction that will lead to maximum estimated increase in the lower bound of the subproblems is a good strategy. Such estimates are made primarily in one of two ways. *Strong branching* consists of making the estimates by partially solving each subproblem created by branching for each candidate variable disjunction. *Pseudo-cost branching* consists of estimating the change on the basis of the actual change that occurred when the candidate disjunction was previously imposed (in some other subproblem). Recently, Achterberg et al. [2005]

showed empirically that using a hybrid approach, called *reliability branching*, yields better results in practice than either of above two approaches used alone.

The study of branching on general disjunctions is not new either and has been previously recognized as an important aspect of the theory of integer programming. In their survey, Aardal and Eisenbrand [2004] discussed the fact that when the dimension is fixed, polynomial time algorithms for solving integer programs can be obtained by branching on general disjunctions obtained by determining the so-called *thin directions* of the feasible region. These polynomial time algorithms are derived from the seminal work of Lenstra [H.W. Lenstra, 1983] and its extensions. It has also been shown, for instance by Krishnamoorthy and Pataki [2006], that certain specific problems can be solved "easily," if one branches on some particular general disjunction. On the other hand, few heuristics have been proposed that enhance computational performance of standard solvers by using general branching disjunctions. Fischetti and Lodi [2003] proposed a local branching heuristic that uses a general disjunction for branching such that one of the branches has a small feasible region but is more likely to contain feasible solutions with small objective function values. Owen and Mehrotra [2001] used a greedy heuristic to generate branching disjunctions with coefficients in $\{0, 1, -1\}$. Karamanov and Cornuéjols [2007] suggested branching using disjunctions that could be used for generating Mixed Integer Gomory cuts in the branch-and-cut algorithm. Some general branching disjunctions have also been shown to be useful for problems with specific structures like special ordered sets [Beale and Tomlin, 1970].

The remainder of the paper is organized as follows. In Section 2, we present two different criteria by which to select a branching disjunction and describe how to solve the problem of determining the optimal general disjunction with respect to these criteria. In Section 3, we analyze the results of computational experiments applying the methods from Section 2. In Section 4, we present our conclusions and indicate directions for future work in this area.

2 Selecting Branching Disjunctions

As previously described, selecting a branching disjunction based on its global effect is likely to be extremely difficult and we must therefore resort to more myopic (though still not theoretically efficient) selection procedures. The two criteria we use here to evaluate a branching disjunction are (i) lower bound improvement achieved after branching and (ii) width of $\mathcal{P}$ in the direction of the disjunction. The problem of finding an optimal general branching disjunction according to each of these criteria is formulated in the following two sections. It is known from the results of Sebő [1999] and our recent work [Mahajan and Ralphs, 2008] that the problem of optimizing over the set of

general branching disjunctions with either of the above criteria is $\mathcal{NP}$-hard, even when the set of disjunctions is restricted in various ways.

2.1 Branching to Maximize Lower Bound

As previously mentioned, experiments by Linderoth and Savelsbergh [1999] and Achterberg et al. [2005] provided empirical evidence that selecting variable disjunctions on the basis of estimated increase in the lower bound after such a branching could result in a reduction in the number of subproblems solved. Therefore, we base our first criteria for choosing branching disjunctions on the same principle. The procedure is based on detecting infeasibility of the subproblems resulting from imposition of the branching disjunction, along with (possibly) an inequality requiring a certain target increase in the lower bound.

First, consider the integer program (P) and assume that $\mathcal{P}$ is not empty (otherwise, the problem is easy to solve). Let $(\hat{\pi}, \hat{\pi}_0) \in \mathbb{Z}^d \times \{0\}^{n-d} \times \mathbb{Z}$ be a disjunction that is used to branch after (P^{LP}) is solved. Then the LP relaxations associated with the two partitions created after branching are of the form

$$\begin{array}{l} \min cx \\ \text{subject to:} \\ \quad Ax \geq b \\ \quad \hat{\pi}x \leq \hat{\pi}_0 \end{array} \quad \text{and} \quad \begin{array}{l} \min cx \\ \text{subject to:} \\ \quad Ax \geq b \\ \quad \hat{\pi}x \geq \hat{\pi}_0 + 1. \end{array} \tag{1}$$

Now, consider the related linear programs

$$\begin{array}{l} z_L^* = \min \hat{\pi}x \\ \text{subject to:} \\ \quad Ax \geq b \end{array} \quad \text{and} \quad \begin{array}{l} z_R^* = \min -\hat{\pi}x \\ \text{subject to:} \\ \quad Ax \geq b. \end{array} \tag{2}$$

The programs (1) are infeasible if and only if $z_L^* > \hat{\pi}_0$ and $z_R^* > -(\hat{\pi}_0 + 1)$. The dual of the programs (2) can be written as

$$\begin{array}{l} z_L^* = \max pb \\ \text{subject to:} \\ \quad pA = \hat{\pi} \\ \quad p \geq 0 \end{array} \quad \text{and} \quad \begin{array}{l} z_R^* = \max qb \\ \text{subject to:} \\ \quad qA = -\hat{\pi} \\ \quad q \geq 0, \end{array} \tag{3}$$

respectively. By imposing the requirement that $z_L^* > \hat{\pi}_0$ and $z_R^* > -(\hat{\pi}_0 + 1)$ and then combining the two dual formulations (3), one can conclude that the LPs (1) are both infeasible if and only if the system

$$\begin{aligned} pA - \pi &= 0 \\ qA + \pi &= 0 \\ pb - \pi_0 &\geq \delta \\ qb + \pi_0 &\geq \delta - 1 \\ p &\geq 0 \\ q &\geq 0 \\ (\pi, \pi_0) &\in \mathbb{Z}^{n+1}, \end{aligned} \tag{4}$$

has a solution for some $\delta > 0$ and with $\pi = \hat{\pi}, \pi_0 = \hat{\pi}_0$.

A sequence of MILPs of the form (4) can now be solved in order to find a branching disjunction whose imposition maximizes the resulting lower bound as follows. Suppose it is desired to increase the lower bound resulting from imposition of the branching disjunction to some value exceeding a given target z_l. This is equivalent to requiring that both the following system of inequalities be infeasible.

$$\begin{array}{lll} Ax \geq b & & Ax \geq b \\ \hat{\pi}x \leq \hat{\pi}_0 & \text{and} & \hat{\pi}x \geq \hat{\pi}_0 + 1 \\ cx \leq z_l & & cx \leq z_l \end{array} \tag{5}$$

Observe that dropping the branching constraints makes the systems (5) feasible, as long as $z_l > z_{LP}$. Using the approach described above, the problem of finding a suitable $(\hat{\pi}, \hat{\pi}_0)$ may now be written as that of finding a feasible solution to the system

$$\begin{aligned} pA - s_L c - \pi &= 0 \\ pb - s_L z_l - \pi_0 &\geq \delta \\ qA - s_R c + \pi &= 0 \\ qb - s_R z_l + \pi_0 &\geq \delta - 1 \\ p, s_L, q, s_R &\geq 0 \\ \pi \in \mathbb{Z}^n, \pi_0 &\in \mathbb{Z}. \end{aligned} \tag{6}$$

The lower bound obtained after solving the LP relaxations (1) can be increased to at least z_l by imposing the branching disjunction $(\hat{\pi}, \hat{\pi_0})$ if and only if $\pi = \hat{\pi}, \pi_0 = \hat{\pi}_0$ is a feasible solution to the system (6) for some $\delta > 0$. Note the similarity in the formulations (6) and (4). If there is a feasible solution to (6) with $s_L = s_R = 0$, then (4) is also feasible and consequently, imposition of the corresponding branching disjunction will make the LPs related to each member of partition infeasible.

If one treats z_l as a variable in formulation (6), then it becomes a nonlinear program because of the presence of bilinear terms $s_L z_l$ and $s_R z_l$. Hence, it is not straightforward to get the maximum value of z_l from this formulation. We overcome this difficulty by solving a sequence of parametric (feasibility)

MILPs of the form (6) by treating z_l as a fixed parameter and choosing a suitable value for δ. By doing a binary search over a range of values for z_l and solving (6) in each iteration of the search, one can obtain the maximum value of the lower bound up to a desired level of accuracy. Additionally, if x^* is known to be a fractional optimal solution of the LP relaxation of the original problem (P), then the constraint $\pi_0 < \pi x^* < \pi_0 + 1$ may optionally be added to formulation (6).

2.2 Branching on Thin Directions

The second criterion by which we judge a branching disjunction (π, π_0) is by the width of $\mathcal{P}$ in the direction π, defined to be $\max_{x,y\in\mathcal{P}} \pi y - \pi x$. Intuitively, a branching disjunction with small associated width should be effective because it is likely that the volume of the union of the feasible regions of the subproblems resulting from imposition of such a disjunction will be significantly smaller than that of the polyhedron $\mathcal{P}$. For a polytope Q, the minimum width in the direction of any general branching disjunction is called the *integer width* and is defined to be

$$w(Q) = \min_{\pi} \max_{x,y\in\mathcal{P}} (\pi y - \pi x) \;\; s.t. \;\; \pi \in \mathbb{Z}^d \times \{0\}^{n-d}, \pi \neq \mathbf{0}.$$

Sebő [1999] showed that for a given polytope Q, the problem of determining whether $w(Q) \leq 1$ is $\mathcal{NP}$-complete, even when Q is a simplex. It is also known, from a result of Banaszczyk et al. [1999], that if Q is empty, then $w(Q) \leq Cn^{\frac{3}{2}}$, where C is a constant. Derpich and Vera [2006] tried to approximate the direction of the minimum integer width in order to assign priorities for branching on variables. They showed that the number of subproblems can be reduced when using this heuristic approach. Aardal and Eisenbrand [2004] discussed the fact that branching on thin directions leads to polynomial time algorithms for integer programs when the dimension is fixed. Hence, such branching directions seem empirically to be useful in reducing the number of subproblems to be examined during the solution procedure.

For a fixed $\hat{\pi}$, the dual of the LP

$$\begin{aligned} &\max \hat{\pi} y - \hat{\pi} x \\ &\text{subject to:} \\ &\quad Ax \geq b \\ &\quad Ay \geq b, \end{aligned} \tag{7}$$

can be written as

$$\begin{aligned} &\min -qb - pb \\ &\text{subject to:} \\ &\quad pA - \hat{\pi} = 0 \\ &\quad qA + \hat{\pi} = 0 \\ &\quad p, q \geq 0. \end{aligned} \tag{8}$$

Thus, the problem of finding $w(\mathcal{P})$ can be equivalently expressed as the program

$$\begin{aligned} &\min -qb - pb \\ &\text{subject to:} \\ &\quad pA - \pi = 0 \\ &\quad qA + \pi = 0 \\ &\quad p, q \geq 0 \\ &\quad \pi \in \mathbb{Z}^n \times \{0\}^{n-d}, \pi \neq \mathbf{0}. \end{aligned} \tag{9}$$

Note that if there exists a feasible solution to formulation (4) described in the previous section, then $w(\mathcal{P}) < 1$. However, the converse is not true. Furthermore, if (P) does not have any continuous variables (i.e., if $d = n$) and if $\mathcal{P}$ is not full dimensional then $w(\mathcal{P}) = 0$. In such a case, one may end up obtaining the same solution from the formulation (9) for each subproblem. In order to overcome this difficulty, we modified the formulation to

$$\begin{aligned} &\min -qb - pb \\ &\text{subject to:} \\ &\quad pA - \pi = 0 \\ &\quad qA + \pi = 0 \\ &\quad \pi_0 + \delta \leq \pi x^* \leq \pi_0 + 1 - \delta \\ &\quad p, q \geq 0 \\ &\quad \pi \in \mathbb{Z}^n \times \{0\}^{n-d} \\ &\quad \pi_0 \in \mathbb{Z}, \end{aligned} \tag{10}$$

where x^* is an optimal solution to the current LP relaxation and δ is a suitably small constant. The formulation (10) is only an approximation to finding the integer width of $\mathcal{P}$. However, it ensures that x^* violates the generated disjunction and also that $\pi \neq \mathbf{0}$.

3 Computational Experiments

In order to test the effect of selecting branching disjunctions using the formulations presented in the previous section, we performed a sequence of experiments using ILOG CPLEX 10.2 with the default selection method for branching disjunctions replaced by the ones previously described. Since our goal was only to discern the effectiveness of employing the disjunctions and not to test the efficiency of the method for determining them, our measure of effectiveness was reduction in total number of subproblems required to solve each instance. Thus, we are ignoring the time required to find the branching disjunctions, which was substantial in some cases.

Initial experiments were carried out on 91 instances selected from MIPLIB 3.0 [Bixby et al., 1998], MIPLIB 2003 [Achterberg et al., 2006], and the Mittelmann test set [Mittelmann, 2008]. The initial set was then reduced to 30 representative instances in order to complete experiments in reasonable time. Table 1 shows the size of these instances. The branching disjunctions were imposed using the callback functions provided with the CPLEX callable library. All experiments were run on 64-bit machines, each with 16GB RAM, 8 1.86GHz cores and 4MB cache. In all experiments, the best known objective function value was provided as upper bound to the solver to ensure that the solution procedure was not affected by the order in which subproblems were solved or other extraneous factors related to improvement in the upper bound.

Table 1 Number of constraints, variables, integer (including binary) variables and binary variables in the 30 instances used in experiments.

Instance	Cons	Vars	Ints	Bins	Instance	Cons	Vars	Ints	Bins
10teams	231	2025	1800	1800	mod008	6	319	319	319
aflow30a	479	842	421	421	neos6	1037	8768	8340	8340
bell3a	123	133	71	39	nug08	913	1632	1632	0
blend2	274	353	264	231	nw04	36	87482	87482	87482
egout	98	141	55	55	p0548	176	548	548	548
fiber	363	1298	1254	1254	pp08aCUTS	246	240	64	64
flugpl	18	18	11	0	qnet1	503	1541	1417	1288
gen	780	870	150	144	qnet1_o	456	1541	1417	1288
gesa2	1392	1224	408	240	ran10x26	297	520	260	260
gesa2_o	1248	1152	672	336	ran12x21	286	504	502	502
gt2	29	188	24	0	ran13x13	196	338	169	169
harp2	112	2993	2993	2993	rout	291	556	315	300
khb05250	101	1350	24	24	stein45	331	45	45	45
l152lav	97	1989	1989	1989	vpm1	234	378	168	168
lseu	28	89	89	89	vpm2	234	378	168	168

In the first experiment, a pure branch-and-bound procedure was used—other advanced techniques such as cutting planes, heuristics and probing were

disabled. This allowed us to observe the effects of branching in isolation from the mitigating effects of applying these additional techniques. In the first experiment, a sequence of MILPs of the form (6) were solved to determine the disjunction yielding the maximum increase in lower bound. During initial testing, we concluded that optimizing over the entire set of general branching disjunctions was too time-consuming, as the MILPs (6) were sometimes extremely difficult to solve. We therefore imposed the following limitations for all tests.

1. π was restricted to the set $\{-M, -M+1, \ldots, M\}^n$. $M = 1$ was used in the first experiment and higher values were tried in other experiments.
2. Each π_i was replaced with two non negative variables substituting $\pi_i = \pi_i^+ - \pi_i^-, \pi_i^+, \pi_i^- \in [0, M]$. Such a transformation was used in order to make it easier for the solver to find heuristic solutions to the MILP formulation.
3. The constraint $\sum_{i=1}^n |\pi_i| \leq k$ was introduced to further restrict the search space. k was set to 2, 5, 10, 15 and 20 in different experiments.
4. A time limit of t seconds was imposed for solving any one MILP for selecting a branching disjunction. Additionally, a limit of $8t$ seconds was imposed on the time allowed to be spent in total on selecting any single branching disjunction. In the first experiment, t was set to 1000. Values of 50 and 100 were used in later experiments.
5. A total time limit of 20 hours was imposed for solving each instance. After 18 hours, only variable disjunctions were considered so that the problem could be solved to completion in the remaining two hours.

In case the search for a branching disjunction failed (because of time limits or because no solution was found), branching was carried out by considering variable disjunctions. Since it was not known how the selection rule of CPLEX works, the LP relaxations of the subproblem resulting from the imposition of each candidate variable disjunction were solved explicitly in order to determine the optimal variable disjunction according to the criteria of maximum increase in lower bound. In cases where it was found that there was no variable disjunction whose imposition resulted in an increase in the lower bound, the default variable branching scheme of CPLEX was invoked. The number of subproblems solved when branching on general disjunctions was compared against that when branching only on variable disjunctions.

The number of subproblems generated during solution of each instance in the first experiment is shown in Table 2. N_k denotes the number of subproblems created when the search was restricted by addition of the constraint $\sum_{i=1}^n |\pi_i| \leq k$. Thus, N_1 denotes the number of subproblems when branching was done using only variable disjunctions (by selecting a variable disjunction after solving the resulting LP relaxations explicitly, as described above). The value r_k is defined to be $\frac{N_1}{N_k}$. Even though the experiments for Table 2 were carried out with 91 instances, only results for the 30 selected for further investigation are reported, since other instances showed similar results.

For all remaining experiments, the performance profiles of Dolan and Moré [2002] are used to display compactly, the results comparing number of subproblems solved in various experiments. A point (α, β) in such a plot indicates that a fraction β of all instances required less than α times the number of subproblems required in the experiment achieving the lowest total overall. Figure 1(a) shows a performance profile for the data in Table 2.

In the next two experiments, the time limit t imposed on the solution of each MILP was reduced to 100 seconds and 50 seconds, respectively. This was done to determine whether good branching disjunctions could still be found in a shorter amount of time. Figures 1(b) and 1(c) show the performance profile when t was fixed and k was varied.

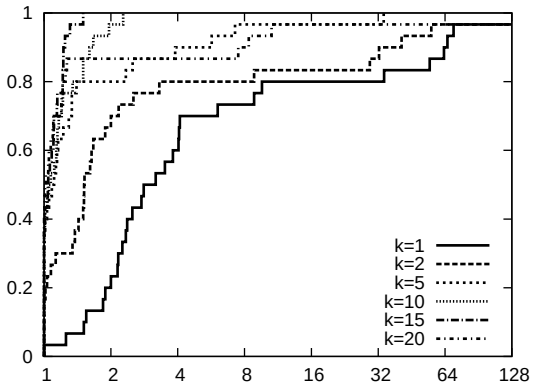
The experiments described so far show that branching on disjunctions that maximize the subsequent lower bound increase does in fact lead to a significant reduction in the number of subproblems required to be solved. In general, the number of subproblems is also reduced when the set of disjunctions considered is larger (i.e., the number of non-zeros allowed in the vector π is increased).

Figures 2(a)-2(e) show the effect of time spent in selecting a branching disjunction when k is fixed. In general, when k is small, additional time spent selecting a disjunction pays a bigger dividend than when k is large. Figure 2(d) shows that when $k = 15$ the number of subproblems solved does not vary much as t is increased. When k is set to 20, the performance with $t = 50$ is nearly equivalent to that with $t = 1000$. One possible explanation is that for large values of k, if a feasible solution to the branching disjunction selection problem is not found quickly, then it is unlikely that a solution will be found even after substantial additional search time. Thus, even though branching on disjunctions that increase the lower bound appears promising, the problem of selecting disjunctions becomes increasingly difficult with the number of nonzero coefficients that are allowed in the description. This seems to be the case for the instance vpm1 in particular (see Table 2)—when k is changed from 15 to 20, the number of subproblems goes up from 20 to 5929, presumably because the branching disjunction selection problem becomes so difficult that only a few effective disjunctions are found within the time limit.

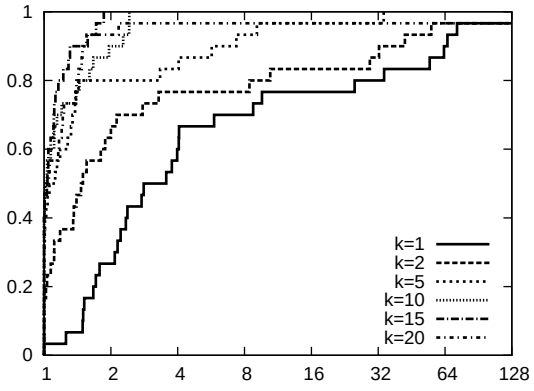
In the next experiment, cutting planes were enabled to see how the branching disjunction selection method would perform in a branch-and-cut algorithm. In general, introduction of cutting plane generation should be expected to reduce the total number of subproblems. The default settings of CPLEX were used for cut generation, with the exception that MIR, Gomory, and flow (cover and path) cuts were disabled because the presence of these cuts caused numerical difficulties while solving some of the associated branching disjunction selection problems. Figure 3(a) shows the effect of adding cuts when $t = 100$ seconds and k has values 1, 2, and 5. It shows that enabling cuts increases the performance of the solver significantly, even when branching on general disjunctions is used. Figure 3(b) shows how the performance varies when cuts are enabled and k is varied from 1 to 20. Figure 1(b) shows

Table 2 Number of nodes (N_i) in branch and bound tree and the ratio $r_i = \frac{N_1}{N_i}$ for selected instances when $t = 1000$ seconds. The criterion for selecting the branching disjunction is to maximize the lower bound.

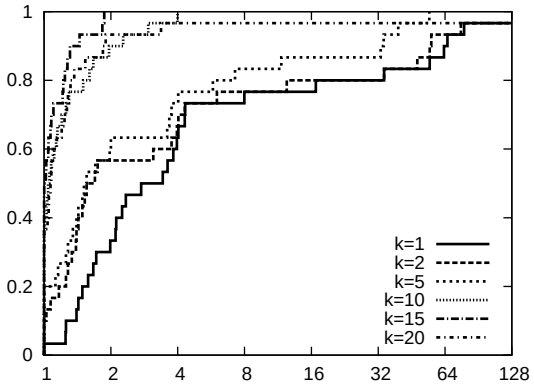
Instance	N_1	N_2	r_2	N_5	r_5	N_{10}	r_{10}	N_{15}	r_{15}	N_{20}	r_{20}
10teams	115	106	1.08	28	4.11	18	6.39	12	9.58	12	9.58
aflow30a	36634	19408	1.89	19485	1.88	20388	1.8	24112	1.52	20271	1.81
bell3a	16387	14377	1.14	8771	1.87	588	27.87	259	63.27	259	63.27
blend2	304	251	1.21	231	1.32	188	1.62	165	1.84	209	1.45
egout	2246	1044	2.15	554	4.05	572	3.93	676	3.32	558	4.03
fiber	18412	7676	2.4	7612	2.42	3039	6.06	3358	5.48	3324	5.54
flugpl	394	176	2.24	6	65.67	10	39.4	6	65.67	6	65.67
gen	100	100	1	100	1	100	1	100	1	100	1
gesa2	33526	24433	1.37	21664	1.55	21849	1.53	21849	1.53	21778	1.54
gesa2_o	98550	24777	3.98	24435	4.03	24661	4	24661	4	24661	4
gt2	340	10	34	10	34	12	28.33	10	34	12	28.33
harp2	432010	157377	2.75	174656	2.47	183306	2.36	174454	2.48	179130	2.41
khb05250	738	606	1.22	594	1.24	588	1.26	614	1.2	618	1.19
l152lav	60	40	1.5	32	1.88	28	2.14	34	1.76	30	2
lseu	4058	2365	1.72	226	17.96	78	52.03	58	69.97	58	69.97
mod008	2840	1678	1.69	296	9.59	102	27.84	68	41.76	52	54.62
neos6	5989	2131	2.81	2131	2.81	2131	2.81	2131	2.81	2131	2.81
nug08	14	6	2.33	4	3.5	6	2.33	6	2.33	5	2.8
nw04	30	24	1.25	16	1.88	12	2.5	12	2.5	12	2.5
p0548	1050	500	2.1	466	2.25	566	1.86	565	1.86	565	1.86
pp08aCUTS	1301300	486340	2.68	147271	8.84	166943	7.79	168905	7.7	231527	5.62
qnet1	42	30	1.4	24	1.75	20	2.1	22	1.91	18	2.33
qnet1_o	154	126	1.22	94	1.64	77	2	80	1.93	92	1.67
ran10x26	68449	34693	1.97	23309	2.94	24716	2.77	23704	2.89	21520	3.18
ran12x21	494558	280551	1.76	219967	2.25	208948	2.37	225980	2.19	212910	2.32
ran13x13	124716	87495	1.43	74699	1.67	57825	2.16	66008	1.89	58789	2.12
rout	219322	79399	2.76	65201	3.36	61806	3.55	61226	3.58	57673	3.8
stein45	31086	31177	1	21238	1.46	20594	1.51	20601	1.51	20601	1.51
vpm1	263111	40952	6.42	145	1814.56	32	8222.22	20	13155.55	5929	44.38
vpm2	273994	145152	1.89	77504	3.54	67014	4.09	69515	3.94	73687	3.72



(a) $t = 1000$ seconds

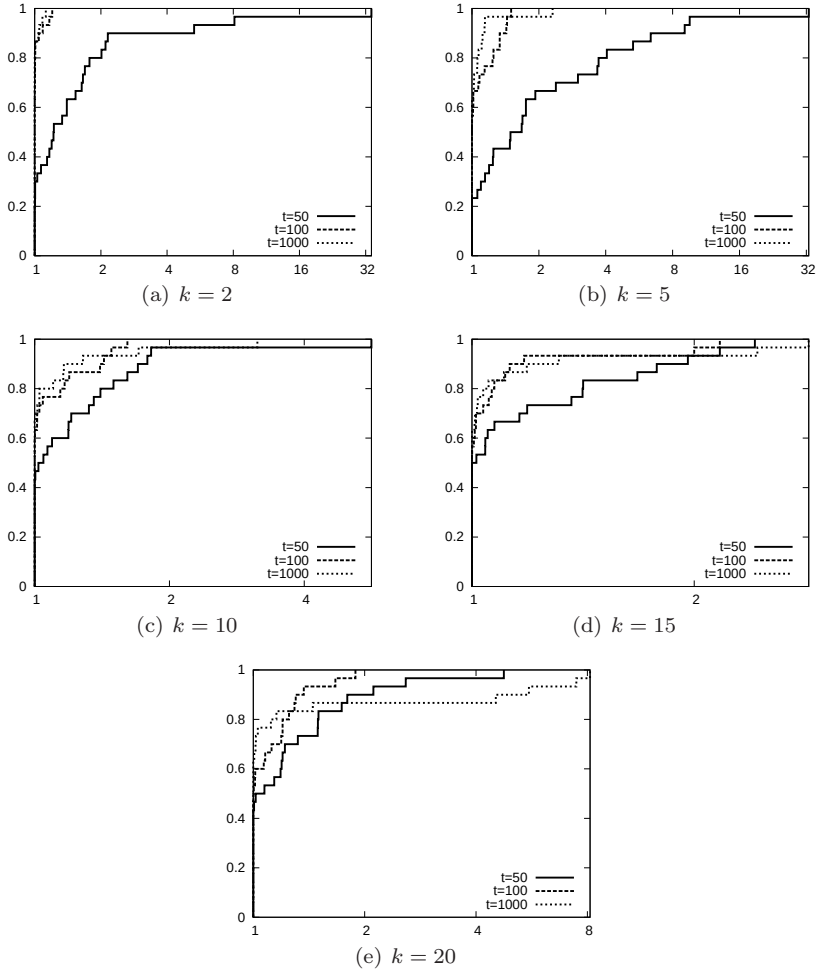


(b) $t = 100$ seconds



(c) $t = 50$ seconds

Fig. 1 Performance profile for number of subproblems when t is fixed and k is varied.



(a) $k = 2$ (b) $k = 5$

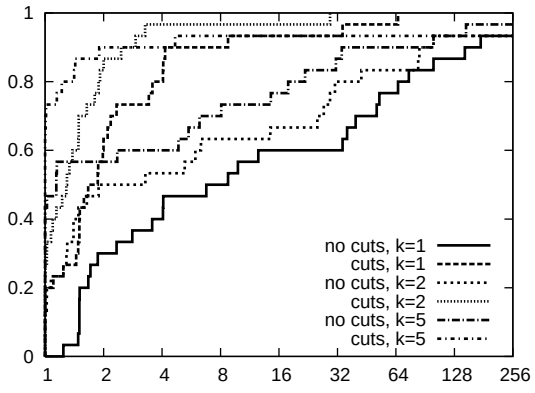
(c) $k = 10$ (d) $k = 15$

(e) $k = 20$

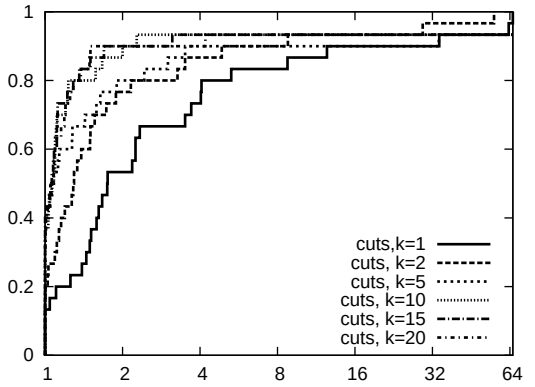
Fig. 2 Performance profile for number of subproblems when k is fixed and t (in seconds) is varied.

that, in the absence of cuts, around 80% of instances required at least half as many subproblems when branching on general disjunctions. When the cuts were enabled, this fraction dropped to 50%. So the effect of branching on general disjunctions is substantial even when the cuts are enabled, though it is not as dramatic.

To see the effect of increasing M, we performed one experiment with $M = 10, k = 15, t = 100$. Figure 4(a) shows a comparison of performance of this test against the others. The performance seems to be slightly worse than when $M = 1$. However, it could not be established whether this was due



(a) $t = 100$ seconds

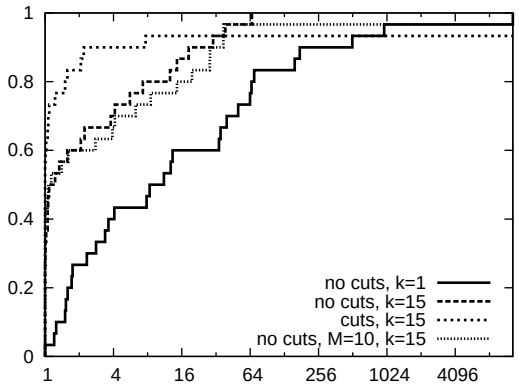


(b) $t = 100$ seconds

Fig. 3 Performance profile for number of subproblems when cuts are added to the original problem. t is fixed and k is varied.

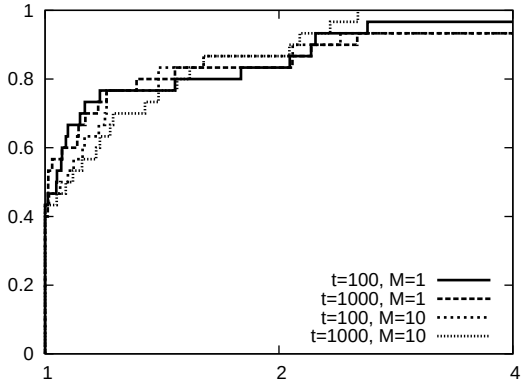
to larger coefficients in some of the disjunctions or because of the increased difficulty of the MILPs used to identify the disjunction. A similar experiment was carried with $M = 10, k = 15, t = 1000$ to see the effects for the case when more time was spent in finding disjunctions with $M > 1$. Figure (5(a)) shows that there are no considerable effects from spending more time or changing M. These experiments seem to suggest that k is probably a more important parameter than either t or M.

Finally, we experimented with selecting a branching disjunction along a "thin" direction by solving the formulation (10). Additional constraints, as described for the criteria of maximizing lower bound above, were also added. Figure 6(a) compares the number of subproblems solved when branching on "thin" directions with other experiments. The performance is seen to be com-



(a) $t = 100$ seconds

Fig. 4 Performance profile to compare the effect of branching for maximum lower bound when M is increased to 10.

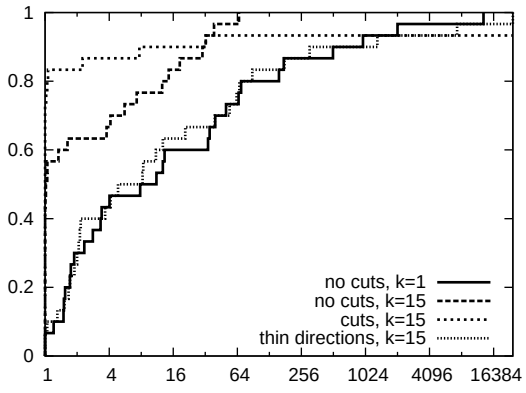


(a) $t = 100$ seconds

Fig. 5 Performance profile to compare the effects of changing t when M in increased.

parable to that of branching on variable-disjunctions. One plausible reason why branching along thin directions did not perform as well as other criteria might be that most of the integer constrained variables in the test set were binary variables. For such problems, the integer width of the polytope associated with the LP relaxation of a subproblem is at most one. Furthermore, there are typically a number of directions along which the width is one. So, for the case when the minimum width of the polytope is one, the formulation (8) selects any one of the many possible directions arbitrarily. One way to overcome this problem would be to resort to other criteria when the mini-

mum width is found to be one. However, we have not yet pursued this line of research.



(a) $t = 100$ seconds

Fig. 6 Performance profile to compare the effect of branching on "thin" directions against other criteria.

4 Conclusion

In this paper, we considered the use of general disjunctions of the form "$\pi x \leq \pi_0 \vee \pi x \geq \pi_0 + 1$" in branch and bound and branch and cut. We formulated the problem of selecting the optimal such disjunction using two different criteria and reported on the effect of using the associated optimization models to select branching disjunctions within the branch-and-bound framework of the commercial solver CPLEX. The naive approach to formulating and solving the branching disjunction selection problem described herein yielded mixed results. The optimization problems that arose turned out to be extremely difficult to solve using off-the-shelf software. Our experiments have given us many ideas as to how improve the efficiency of solving these problems and also how to develop fast heuristics for obtaining "good" disjunctions quickly. However, this is future work and was not the focus of this initial study.

With regard to the effectiveness of using more general disjunctions, our conclusion is that such an approach, if it can be made efficient, will undoubtedly yield improved solution times. We observed consistent substantial reductions in the number of subproblems required to be solved when using general disjunctions for branching. We therefore conclude that this is a fruitful line of future research, though much thought has to go into how to make solution of

the formulations presented here efficient. Other interesting lines of research concern the development of additional criteria for selection of branching disjunctions and the study of the relationship between disjunctions used for generating valid inequalities and those used for branching. Both these topics have been addressed already to some extent, but certainly deserve further study.

5 Acknowledgement

The computational experiments were undertaken on a cluster provided by High Performance Computing at Lehigh University. CPLEX licences were made available by COR@L Labs at Lehigh University. The authors would also like to thank the reviewers for helpful comments and suggestions.

References

Aardal K, Eisenbrand F (2004) Integer programming, lattices and results in fixed dimenstion. Tech. rep., Probability, Networks and Algorithms

Achterberg T, Koch T, Martin A (2005) Branching rules revisited. Operation Research Letters 33:42–54

Achterberg T, Koch T, Martin A (2006) Miplib 2003. Operations Research Letters 34(4):1–12

Banaszczyk W, Litvak AE, Pajor A, Szarek SJ (1999) The flatness theorem for nonsymmetric convex bodies via the local theory of banach spaces. Mathematics of Operations Research 24(3):728–750

Beale EML, Tomlin JA (1970) Special facilities in general mathematical programming system for non-convex problems using ordered sets of variables. In: Lawrence J (ed) Proceedings of the Fifth International Conference on Operations Research, pp 447–454

Bixby RE, Ceria S, McZeal CM, Savelsbergh MW (1998) An updated mixed integer programming library: Miplib 3. Tech. Rep. TR98-03, Rice University

Derpich I, Vera JR (2006) Improving the efficiency of branch and bound algorithm for integer programming based on "flatness" information. European Journal of Operational Research 174:92–101

Dolan ED, Moré JJ (2002) Benchmarking optimization software with performance profiles. Mathematical Programming 91:201–213

Fischetti M, Lodi A (2003) Local branching. Mathematical Programming 98:23–47, series B

HW Lenstra J (1983) Integer programming with a fixed number of variables. Mathematics of Operations Research 8:538–548

Karamanov M, Cornuéjols G (2007) Branching on general disjunctions, working Paper

Krishnamoorthy B, Pataki G (2006) Column basis reduction and decomposable knapsack problems. Submitted, available at http://www.optimization-online.org/DB_HTML/2007/06/1701.html

Linderoth J, Savelsbergh M (1999) A computational study of search strategies for mixed integer programming. INFORMS Journal on Computing 11:173–187

Mahajan A, Ralphs TK (2008) On the complexity of branching on general hyperplanes for integer programming. Working paper

Mittelmann H (2008) Mixed integer LP problems. Available at http://plato.la.asu.edu/ftp/milp/

Nemhauser GL, Wolsey LA (1988) Integer and Combinatorial Optimization. John Wiley & Sons, Inc.

Owen JH, Mehrotra S (2001) Experimental results on using general disjunctions in branch-and-bound for general-integer linear programs. Computational optimization and applications 20(2)

Sebő A (1999) An introduction to empty lattice simplices. In: Proceedings of the 7th International IPCO Conference on Integer Programming and Combinatorial Optimization, LNCS, pp 400–414