Math. Program., Ser. A (2008) 113:219–240
DOI 10.1007/s10107-006-0049-5

FULL LENGTH PAPER

Optimizing over the split closure

Egon Balas · Anureet Saxena

Received: 20 January 2006 / Accepted: 6 October 2006 / Published online: 15 November 2006

Abstract The polyhedron defined by all the split cuts obtainable directly (i.e. without iterated cut generation) from the LP-relaxation P of a mixed integer program (MIP) is termed the (elementary, or rank 1) split closure of P. This paper deals with the problem of optimizing over the elementary split closure. This is accomplished by repeatedly solving the following separation problem: given a fractional point, say x, find a rank-1 split cut violated by x or show that none exists. Following Caprara and Letchford [17], we formulate this separation problem as a nonlinear mixed integer program that can be treated as a parametric mixed integer linear program (PMILP) with a single parameter in the objective function and the right hand side. We develop an algorithmic framework to deal with the resulting PMILP by creating and maintaining a dynamically updated grid of parameter values, and use the corresponding mixed integer programs to generate rank 1 split cuts. Our approach was implemented in the COIN-OR framework using CPLEX 9.0 as a general purpose MIP solver. We report our computational results on well-known benchmark instances from MIPLIB 3.0 and several classes of structured integer and mixed integer problems. Our computational results show that rank-1 split cuts close more than 98% of the duality gap on 15 out of 41 mixed integer instances from MIPLIB 3.0. More than 75% of the duality gap can be closed on an additional

Research was supported by the National Science Foundation through grant #DMI-0352885 and by the Office of Naval Research through contract N00014-03-1-0133.

E. Balas (✉) · A. Saxena
Tepper School of Business, Carnegie Mellon University, Pittsburgh, PA 15213-3890, USA
e-mail: eb17@andrew.cmu.edu

A. Saxena
e-mail: anureet@cmu.edu

10 instances. The average gap closed over all 41 instances is 72.78%. In the pure integer case, rank-1 split cuts close more than 75% of the duality gap on 13 out of 24 instances from MIPLIB 3.0. On average, rank 1 split cuts close about 72% of the duality gap on these 24 instances. We also report results on several classes of structured problems: capacitated versions of warehouse location, single-source facility location, p-median, fixed charge network flow, multi-commodity network design with splittable and unsplittable flows, and lot sizing. The fraction of the integrality gap closed varies for these problem classes between 100 and 67%. We also gathered statistics on the average coefficient size (absolute value) of the disjunctions generated. They turn out to be surprisingly small.

1 Introduction: the split closure of a MIP

Consider the mixed integer program

$$\min\{cx : Ax \geq b,\ x_j \text{ integer},\ j \in N_1\} \tag{MIP}$$

where A is $m \times n$ rational matrix, $N := \{1, \ldots, n\}$, $N_1 \subseteq N$, and $Ax \geq b$ subsumes the upper bound constraints for variables $j \in N_1$, and $x_j \geq 0\ j \in N$. Let the linear programming relaxation of mixed integer program (MIP) be

$$\min\{cx : x \in P\},$$

where $P := \{x \in \mathbb{R}^n : Ax \geq b\}$, and let $P_I := \text{conv}(P \cap \{x : x_j \text{ integer},\ j \in N_1\})$.

Given any integer vector $(\pi, \pi_0) \in \mathbb{Z}^{n+1}$ such that $\pi_j = 0, j \in N_2 := N \setminus N_1$, the *split disjunction*

$$\pi x \leq \pi_0 \quad \vee \quad \pi x \geq \pi_0 + 1 \tag{LP}$$

is obviously satisfied by any x with integer components in N_1. Any valid inequality derived from such a disjunction is called a *split cut*. More generally, we will call a split cut any valid inequality for P_I derived from a disjunction of the form

$$\begin{pmatrix} Ax \geq \ \ b \\ -\pi x \geq -\pi_0 \end{pmatrix} \vee \begin{pmatrix} Ax \geq b \\ \pi x \geq \pi_0 + 1 \end{pmatrix}. \tag{$D(\pi, \pi_0)$}$$

Split cuts derived from $D(\pi, \pi_0)$ directly, i.e. without iterating the cut derivation process, will be called *elementary*, or rank 1, split cuts. The general form of such a cut is $\alpha x \geq \beta$ (see [6,7]), where

$$\begin{aligned} \alpha &= uA - u_0\pi \\ \alpha &= vA + v_0\pi \\ \beta &= ub - u_0\pi_0 \\ \beta &= vb + v_0(\pi_0 + 1) \\ u, u_0, v, v_0, &\geq 0. \end{aligned} \tag{1.1}$$

The polyhedron defined by the set of elementary split cuts for all integer vectors (π, π_0) with $\pi_j = 0$, $j \in N_2$, will be called the *elementary split closure of P*, or, when there is no danger of confusion, simply the *split closure of P*, denoted $\mathcal{C}$.

Split cuts (the term is due to [19]) are a special class of disjunctive cuts [6] known in the more recent literature as lift-and-project cuts [7]. For the connection between split cuts, lift-and-project cuts and mixed integer Gomory cuts, see [11,17,20].

2 How to optimize?

Optimizing over the elementary closure of any class of cutting planes poses the challenge that as cuts are generated and added to the linear programming relaxation to redefine its optimum, they cannot be used to generate further cuts. Recently, Fischetti and Lodi [23] have pointed out a way to do this for the case of the elementary Chvatal closure of a pure integer program. In this paper we take a similar approach to optimizing over the split closure of a mixed integer program. Work in this vein is also reported in [15,16,21]. Caprara and Letchford [17] have formulated the problem of optimizing over the split closure as a mixed integer nonlinear program and have shown that the separation problem for split cuts is strongly $\mathcal{NP}$-complete.

Given a MIP, by optimizing over the split closure we mean finding $\min\{cx : x \in \mathcal{C}\}$. We accomplish this by an iterative procedure, which alternates between solving a Master Problem and a Separation Problem. The Master Problem is a linear program of the form

$$\begin{aligned} \min\ & cx \\ & Ax \geq b \\ & \alpha^t x \geq \beta^t, \quad t \in T \end{aligned} \tag{MP}$$

where $Ax \geq b$ is the constraint set of the linear programming relaxation P, and $\alpha^t x \geq \beta^t$, $t \in T$, are the elementary split cuts generated so far. If the optimal solution of (MP) at some iteration is $\hat{x}$, the Separation Problem is of the form

$$\begin{aligned} \min\ & \alpha\hat{x} - \beta \\ & \alpha = uA - u_0\pi \\ & \alpha = vA + v_0\pi \\ & \beta = ub - u_0\pi_0 \\ & \beta = vb + v_0(\pi_0 + 1) \\ & 1 = u_0 + v_0 \\ & u, u_0, v, v_0 \geq 0 \\ & (\pi, \pi_0) \text{ integer}, \quad \pi_j = 0, \ \ j \in N_2 \end{aligned} \tag{SP}$$

The constraint set of (SP) is obtained from (1.1) by adding the normalization constraint $u_0 + v_0 = 1$, and allowing (π, π_0) to vary over all integer $(n+1)$-vectors. An optimal solution to (SP) either yields a cut $\alpha x \geq \beta$ violated by $\hat{x}$ (if $\alpha\hat{x} - \beta < 0$) or else proves that $\hat{x} \in \mathcal{C}$.

Since $\mathcal{C}$ is known to be a polyhedron [19], this procedure finds the minimum of cx in a finite number of iterations if the procedure used for solving (SP) satisfies a certain technical condition (see Sect. 3). Solving (MP) at each iteration poses no problem. On the other hand (SP) is a mixed integer nonlinear program involving products of integer and continuous variables. However, it is possible to get rid of these cross-products and to restate the problem as a parametric mixed integer linear program with a single parameter in the objective function and the right hand side. This can be accomplished by eliminating α, β and using the equation $u_0 + v_0 = 1$ to make all coefficients of (π, π_0) equal to 1, -1 or 0. The resulting problem is

$$\begin{aligned} \min \quad & u(A\hat{x} - b) - u_0(\pi\hat{x} - \pi_0) \\ & uA - vA - \pi = 0 \\ & -ub + vb + \pi_0 = u_0 - 1 \\ & u, v \geq 0, \ \ \pi, \pi_0 \text{ integer}, \quad \pi_j = 0, \ \ j \in N_2 \\ & 0 \leq u_0 \leq 1 \end{aligned} \tag{2.1}$$

It is possible to further simplify this formulation by eliminating (π, π_0) from the objective function. This is not necessarily useful computationally, but it yields some insights. Define $\hat{s} := A\hat{x} - b$.

Proposition 2.1 *The objective function of (2.1) is equivalent to*

$$(v_0 u + u_0 v)\hat{s} - u_0 v_0. \tag{2.2}$$

Proof

$$\begin{aligned} \alpha\hat{x} - \beta &= u(A\hat{x} - b) - u_0(\pi\hat{x} - \pi_0) \\ &= u\hat{s} - u_0(\pi\hat{x} - \pi_0) \\ \alpha\hat{x} - \beta &= v(A\hat{x} - b) + v_0(\pi\hat{x} - \pi_0 - 1) \\ &= v\hat{s} + v_0(\pi\hat{x} - \pi_0) - v_0 \end{aligned}$$

Multiplying by v_0 the first expression for $\alpha\hat{x} - \beta$, and by u_0 the second expression for $\alpha\bar{x} - \beta$, and adding the two expressions yields the value (2.2) for $\alpha\bar{x} - \beta$.

Now, letting $u_0 = \theta$, $v_0 = 1 - \theta$, we obtain a parametric mixed integer linear program with the scalar parameter θ in the objective function and right hand side.

$$\begin{aligned} \min z(\theta) = (1-\theta)u\hat{s} + \theta\, v\hat{s} - \theta(1-\theta) & \\ uA - vA - \pi & = 0 \\ -ub + vb + \pi_0 & = \theta - 1 \qquad \text{(PMILP)} \\ u, v \geq 0,\ \ \pi, \pi_0 \text{ integer}, \pi_j = 0,\ \ j \in N_2 & \\ 0 \leq \theta \leq 1 & \end{aligned}$$

Dropping the integrality constraints on π, π_0 yields as a relaxation a parametric linear program which can be solved by a variant of the simplex algorithm (see for instance Nazareth [25]).

The above formulation immediately yields a lower bound on the value of the optimum:

Proposition 2.2 *For all* θ, $0 \leq \theta \leq 1$, $\min z(\theta) \geq -\theta(1-\theta)$.

Proof All terms of $z(\theta)$ other than $-\theta(1-\theta)$ are nonnegative.

The parametric mixed integer linear program (PMILP) has a certain kind of symmetry. Indeed, let us denote $w := (u, v, \pi, \pi_0)$. Then we have

Proposition 2.3 *Let* $\bar{w}$ *be a feasible solution to PMILP for* $\theta = \bar{\theta}$ $(0 \leq \bar{\theta} \leq 1)$, *and define* $\hat{w}$ *by*

$$\hat{u} := \bar{v}, \quad \hat{v} := \bar{u}, \quad \hat{\pi} := -\bar{\pi} \ \text{ and } \ \hat{\pi}_0 := -\bar{\pi}_0 - 1.$$

Then $\hat{w}$ *is a feasible solution to PMILP for* $\theta = 1 - \bar{\theta}$, *and* $\bar{z}(\bar{\theta}) = \hat{z}(1-\bar{\theta})$.

Proof Substitution of $\hat{w}$ into PMILP shows that it is feasible for $\theta = 1 - \bar{\theta}$ and that $\bar{z}(\bar{\theta}) = \hat{z}(1-\bar{\theta})$.

Corollary 2.4 *For all* θ, $0 \leq \theta \leq 1$, $\min z(\theta) = \min z(1-\theta)$.

The basic properties of the separation problem are summarized in the following.

Theorem 2.5 *The point* $\hat{x}$ *lies in the elementary split closure of* P *if and only if the optimum of PMILP is nonnegative. Furthermore, if* (u, v, π, π_0) *is a feasible* (*not necessarily optimal*) *solution to PMILP for some* θ, $0 < \theta < 1$, *with* $\min z(\theta) < 0$, *then the corresponding* (α, β) *given by* (1.1) (*with* $u_0 = \theta, v_0 = 1-\theta$) *defines a rank 1 split cut* $\alpha x \geq \beta$ *violated by* $\hat{x}$.

Proof Recall that

$$\begin{aligned} z(\theta) &= u\hat{s} - \theta(\pi\hat{x} - \pi_0) \\ &= (uA - \theta\pi)\hat{x} - ub + \theta\pi_0 \\ &= \alpha\hat{x} - \beta. \end{aligned}$$

Thus $\hat{x}$ lies in $\mathcal{C}$ if and only if the minimum of $\alpha\hat{x} - \beta$ over all (α, β) that define a rank 1 split cut is nonnegative. Further, any feasible solution to PMILP defines a split cut $\alpha x \geq \beta$ of rank 1, and if $\alpha\hat{x} - \beta < 0$, then $\hat{x}$ violates $\alpha x \geq \beta$.

3 Solving the separation problem

For practical considerations, we applied several modifications to PMILP, which are listed below.

M1 In spite of the insights provided by the formulation using the objective function (2.2), we found it computationally more convenient to work with the objective function used in (2.1). Also, from practical considerations, we impose (loose) lower and upper bounds on the components of (π, π_0).

M2 From the shape of the objective function $z(\theta)$ it is clear that whenever $\hat{s}_i = 0$ for some i, say $i \in M$, there is a high likelihood of the existence of multiple optimal solutions with differing values u_i for $i \in M$. Some of these solutions may give rise to unnecessarily large cut coefficients. To avoid this phenomenon we modify $z(\theta)$ by replacing $\hat{s}$ with $\bar{s}$ defined by

$$\bar{s}_i := \max\{\sigma, \hat{s}_i\}, \quad \forall i$$

where $\sigma = 0.0001$.

M3 Since from Corollary 2.4 the optimum of PMILP for $\theta = \bar{\theta}$ is the same as for $\theta = 1 - \bar{\theta}$, we restrict the interval of variation of θ from $(0, 1)$ to $(0, \frac{1}{2}]$.

M4 Following the approach used in Balas et al. [8] and all subsequent practical lift-and-project cut generating procedures, we work in the subspace of those variables that are not at one of their bounds in the incumbent solution, and lift the resulting cuts to the full space as in [8].

Parametric mixed integer linear programs can be solved by a branch and bound algorithm for parametric MILP where at every node of the search tree a parametric linear program is used as a relaxation. Since the separation procedure can only generate a finite number of distinct (π, π_0) vectors, if cuts are generated from the basic solutions of (SP) associated with each (π, π_0), their number is finite and the entire procedure terminates in a finite number of steps.

However, such an approach would prevent us from using a state-of-the-art MILP solver like CPLEX or XPRESS. We therefore found it preferable to deparametrize PMILP, and solve it for fixed values of θ as a mixed integer linear program MILP(θ), using CPLEX 9.0. Thus, we create a *parameter grid*, i.e. a grid of values $\theta_1 < \theta_2 < \cdots < \theta_k$, initialized with $\{0, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$, and subsequently enrich it whenever necessary by bisecting an interval, i.e. introducing between θ_t and θ_{t+1} the new grid point $\theta_{t,t+1} := (\theta_t + \theta_{t+1})/2$. Thus rather than solving PMILP to guaranteed optimality, we are approximating the optimum from above; and the denser our grid, the more accurate the approximation.

Any feasible solution (u, v, π, π_0) to MILP(θ) with $z(\theta) < 0$, whether optimal or not, provides a split cut $\alpha x \geq \beta$ of rank 1 violated by $\hat{x}$, along with the disjunction $D(\pi, \pi_0)$ from which $\alpha x \geq \beta$ is generated as a lift-and-project cut.

Formulating the separation problem as a parametric MILP was made possible by the use of the normalization $u_0 + v_0 = 1$. However, it is well known (and confirmed by our experience) that this normalization yields weaker cuts than the normalization

$$ue + ve + u_0 + v_0 = k \tag{3.1}$$

where $e = (1, \ldots, 1)$, used in [10,11]. Therefore, rather than using the cut $\alpha x \geq \beta$ provided directly by the solution, we use the disjunction $D(\pi, \pi_0)$ provided by the latter to derive a lift-and-project cut with the normalization (3.1) with $k = 2|e| + 2$, by solving the corresponding cut generating linear program. Furthermore, we do this every time a new feasible solution to PMILP is discovered.

Solving MILP(θ) for some θ is aimed at finding a cut violated by the current solution $\hat{x}$ by as much as possible. However, any cut violated by $\hat{x}$ may be useful, and so as soon as we find one we store it. Beyond this, we also use two auxiliary devices to tighten the constraint set of MILP(θ) in a way that yields additional cuts. This is legitimate, since any feasible solution to MILP(θ) such that $z(\theta) < 0$, obtained by whatever means, yields a split cut of rank 1. The first such device is along the following lines. If $\bar{x}$ is any vector obtained from $\hat{x}$ by rounding the components $j \in N_1$ to integer values, then $\pi\bar{x} - \pi_0$ must be integer whenever (π, π_0) is. Therefore we may introduce a new integer-constrained variable ζ and impose the constraint $\zeta = \pi\bar{x} - \pi_0$. Since from the equations of PMILP we have that $\pi\bar{x} - \pi_0 = (u - v)(A\bar{x} - b) - (1 - \theta)$, we add the new constraint in the form

$$\zeta = (u - v)(A\bar{x} - b) - (1 - \theta), \quad \zeta \text{ integer}.$$

This increases by 1 the dimension of MILP(θ), but tightens its constraint set. We use this "1-lifting" whenever no new cut was found for a while.

The second device is aimed at diversifying the set of disjunctions by enforcing their partial orthogonality in some directions. To be specific, every time we move to a new grid point we impose the condition $\pi_j = 0, j \in N_1^*$, where N_1^* is the support of a heuristically chosen solution to the set covering problem

$$\min_z \left\{ \sum_{j \in N_1} \min\{\hat{x}_j - \lfloor \hat{x}_j \rfloor, \lceil \hat{x}_j \rceil - \hat{x}_j\} z_j : \sum_{j \in N_1} f(\pi_j^t) z_j \geq 1, \forall t \in T, z_j \in \{0, 1\}, \forall j \in N_1 \right\},$$

where T is the set of disjunctions already discovered ($N_1^* = \emptyset$ if $T = \emptyset$) and f is defined as: $f(k) = 1$ if $k \neq 0$, 0 otherwise. The condition $\pi_j = 0$ $j \in N_1^*$ excludes the disjunctions in T from being rediscovered by PMILP at other grid points. Notice that the procedure is only used if at least one cut has already been discovered during the current iterations, so that its use cannot affect the validity of the separation algorithm.

Since our separation procedure typically generates multiple cuts at each iteration, we use a bound σ on the number of cuts that we accept, and whenever σ is

attained, the iteration is terminated. This bound σ is an increasing function of g, the number of fractional $\hat{x}_j$, $j \in N_1$, with a slope of 1 for $g \leq 75$, and a gradually decreasing slope for $g > 75$. Besides, in order to cope with a rapidly increasing number of cuts, after every three iterations we remove from the master problem all cuts nonbinding for the incumbent solution.

A flow chart of our algorithm is shown in Fig. 1

The separation procedure described above provides a lower bound on the integrality gap closed by the split closure. Because the separation problem is not solved for every possible value of the parameter θ, but only for the values of θ on the grid, our procedure does not provide a guarantee that this lower bound is best possible. However, it can be shown [12] that the average coefficient size of a disjunction needed to cut off the incumbent solution increases with the density of the grid. Hence for a sufficiently dense grid, a split disjunction with very large coefficients would be required to cut off the last solution. As our computational experiments show (see the next section), such a disjunction is unlikely to exist.

4 Computational results

We implemented the separation algorithm (discussed in the previous section) using COIN-OR [18] and CPLEX (version 9.0). The branch-and-bound module of CPLEX was used to generate feasible solutions to MILP(θ), supplemented by the use of COIN-OR's linear programming module for generating cuts from the resulting disjunctions. COIN-OR's linear programming module was also used to solve the master problem (MP). In this section, we describe the computational results of our experiment on a test-bed consisting of several hundred instances from the literature, starting with those of MIPLIB 3.0, and continuing with three categories of structured problems: location, network design and lot

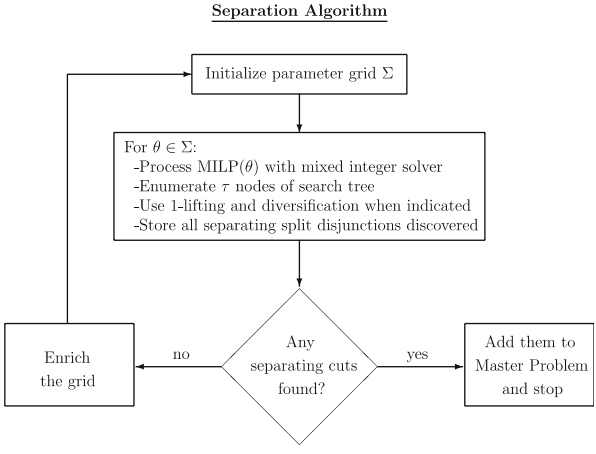

Fig. 1 Flowchart of Separation Algorithm

sizing. In this experiment, our goal was to close as large a portion of the integrality gap as possible, in order to assess the strength of the split closure. This of course has led in many cases to exorbitantly long runs, but our goal was to a large extent attained.

Additional details of our computational experiments are available at the website OSCLIB [26]. In particular, complete lists of the split cuts binding for the final solution to the Master Problem, and the disjunctions giving rise to these cuts, are available for each instance.

4.1 MIPLIB 3.0

We ran our code on all of the instances in MIPLIB 3.0. For each instance, the experiment was terminated when our separation procedure either found $z(\theta) \geq 0$ for all values of θ in the current grid enriched once, or was unable to find a new violated cut within 1 hour from the last one, irrespective of the number of grid points examined. The total running times vary widely with the instance and more information is given in the tables. The percentage of duality gap closed, computed as $100-100\times$(remaining gap/original gap), is an underestimate, as we only have a lower bound on the optimal value over the split closure.

Table 1 reports our results on mixed integer programming instances from MIPLIB 3.0. The second column of the table gives the total number of constraints. The third and fourth column give the number of variables and the number of integer-constrained variables, respectively. The fifth column gives the percentage of integrality gap closed by our procedure, which is a lower bound on the gap closed by optimizing over the elementary split closure. For the sake of comparison, the sixth column gives the lower bound on the percentage of integrality gap closed by the first Chvátal closure of the projection of P onto the subspace of integer-constrained variables (pro-CG), as reported in [16]. The next column gives the number of iterations between the master problem and the separation problem, while the next two columns give the number of cuts generated (total) and those active at termination (binding for the last solution). The last column gives the total running time in seconds. Table 2 summarizes these results. The averages quoted in Table 2 are obtained after excluding instances dsbmip and noswot (duality gap=0) and rentacar (numerical problems).

Table 3 reports our results on all of the *pure integer* programs from MIPLIB 3.0. All of these instances except for gt2, have only binary variables. The second and third columns of the table give the number of constraints and (integer constrained) variables. The fifth column gives the best lower-bound obtained by our code on the integrality gap closed by the elementary split closure. For the sake of comparison, the fifth column gives the best lower bound obtained by Fischetti and Lodi [23] on the gap closed by the first Chvátal closure. The last four columns have the same interpretations as in Table 2. Table 4 summarizes these results. The averages quoted in Table 4 are obtained after excluding the instance enigma (duality gap = 0).

Table 1 Mixed IP instances (MIPLIB 3.0)

Instance	# Constraints	# Variables		% Gap closed by SC	% Gap closed by pro-CG	# Iterations	# Cuts		Time (s)
		Total	Integer				Total	Binding for the last solution	
10teams	230	2,025	1,800	100.00	57.14	2	206	37	90
arki001	782	959	483	83.05	28.04	92	8,464	227	193,536
bell3a	123	133	71	65.35	48.10	31	1,984	26	102
bell5	91	104	58	91.03	91.73	16	800	26	2,233
blend2	274	353	264	46.52	36.40	85	1,020	52	552
dano3mip	3,202	13,873	552	0.22	0.00	10	1,020	153	73,835
danoint	664	521	56	8.20	0.01	7	595	215	147,427
dcmulti	290	548	75	100.00	47.25	20	1,660	213	2,154
dsbmip	1,182	1,886	192	–	–	5	400	64	370
egout	98	141	55	100.00	81.77	8	616	57	18,179
fiber	363	1,298	1,254	99.68	4.83	232	18,792	113	163,802
fixnet6	478	878	378	99.75	67.51	122	11,102	321	19,577
flugpl	23	23	11	100.00	19.19	6	120	23	26
gen	780	870	150	100.00	86.60	6	474	32	46
gesa2	1,392	1,224	408	99.02	94.84	53	4,717	116	22,808
gesa2_o	1,248	1,224	720	99.97	94.93	73	7,154	140	8,861
gesa3	1,368	1,152	384	95.81	58.96	32	3,200	129	30,591
gesa3_o	1,224	1,152	672	95.20	64.53	19	1919	260	6,530
khb05250	101	1,350	24	100.00	4.70	13	494	72	33
markshare1	6	62	50	0.00	0.00	47	564	33	1,330
markshare2	7	74	60	0.00	0.00	73	1,022	40	3,277
mas74	13	151	150	14.02	0.00	27	648	29	1,661
mas76	12	151	150	26.52	0.00	9	198	36	4,172
misc06	820	1,808	112	100.00	0.00	10	340	31	229
mkc	3,411	5,325	5,323	36.16	1.27	84	8,484	803	51,918
mod011	4,480	10,958	96	72.44	0.00	577	18,464	675	86,385
modglob	291	422	98	92.18	0.00	126	7,560	87	1,594
noswot	182	128	100	–	–	243	4,860	33	684
pk1	45	86	55	0.00	0.00	61	1,830	20	55
pp08a	136	240	64	97.03	4.32	45	3,870	127	12,482
pp08aCUTS	246	240	64	95.81	0.68	54	4,374	113	5,666
qiu	1,192	840	48	77.51	10.71	128	9,216	220	200,354
qnet1	503	1,541	1,417	100.00	7.32	62	5,270	113	21,498
qnet1_o	456	1,541	1,417	100.00	8.61	200	4,400	111	5,312
rentacar	6,803	9,557	55	0.00	0.00	0	0	0	–
rgn	24	180	100	100.00	0.00	21	714	39	222
rout	291	556	315	70.70	0.03	367	28,626	124	464,634
set1ch	492	712	240	89.74	51.41	30	3,060	223	10,768
swath	884	6,805	6,724	28.51	7.68	28	2,268	319	2,420
vpm1	234	378	168	100.00	100.00	63	2,142	47	5,010
vpm2	234	378	168	81.05	62.86	41	2,624	111	6,012

For instances dsbmip and nowsot there is no gap between the value of the initial (fractional) LP solution and the optimal value. For instance rentacar the code encountered numerical problems

The support of a split disjunction, $D(\pi, \pi_0)$, is the set of non-zero components of π. In our experiment, we found that most of the split disjunctions discovered during the separation phase had support of size less than 20, irrespective of problem size. Although small support of a disjunction does not imply sparsity of the cut derived from it, the two are not unrelated. The importanceof sparsity

Table 2 Summary of mixed IP instances (MIPLIB 3.0)

Total number of mixed IP instances	41
Average % gap closed by split closure	72.78%
98–100% gap closed	15
75–98% gap closed	10
25–75% gap closed	7
<25% gap closed	6

Table 3 Pure IP instances (MIPLIB 3.0)

Instance	# Constraints	# Variables	% Gap closed by SC	% Gap closed by CG	# Iterations	# Cuts Total	# Cuts Binding for the last solution	Time (s)
air03	124	10,757	100.00	100.00	1	72	3	3
air04	823	8,904	91.23	30.40	128	13,184	602	864,630
air05	426	7,195	61.98	35.30	160	16,480	192	24,156
cap6000	2,176	6,000	65.17	22.50	151	604	32	1,260
enigma	21	100	–	–	167	2,004	4	3,612
fast0507	507	63,009	19.08	5.30	75	4,725	164	304,331
gt2	29	188	98.37	91.00	18	396	66	599
harp2	112	2,993	46.98	49.50	12	720	91	7,671
l152lav	97	1,989	95.20	59.60	541	46,526	80	496,652
lseu	28	89	93.75	93.30	53	1,166	19	32,281
misc03	96	160	51.70	51.00	68	2,720	71	18,359
misc07	212	260	20.11	19.00	83	4,648	184	41,453
mitre	2,054	10,724	100.00	16.20	22	2,288	523	5,330
mod008	6	319	99.98	100.00	21	210	16	85
mod010	146	2,655	100.00	100.00	17	1,190	9	264
nw04	36	87,482	100.00	100.00	14	168	146	996
p0033	16	33	87.42	85.30	27	324	16	429
p0201	133	201	74.93	60.60	40	1600	36	31,595
p0282	241	282	99.99	99.90	127	6,604	69	58,052
p0548	176	548	99.42	62.40	130	10,660	230	9,968
p2756	755	2,756	99.90	42.60	98	4,704	298	12,673
seymour	4,944	1,372	61.52	33.00	69	7,176	413	775,116
stein27	118	27	0.00	0.00	38	1,596	20	8,163
stein45	331	45	0.00	0.00	70	4,900	34	27,624

For instance enigma there is no gap between the value of the initial (fractional) LP solution and the optimal value

Table 4 Summary of pure IP instances (MIPLIB 3.0)

Total number of pure IP instances	24
Average % gap closed by split closure	72.47%
98–100% gap closed	9
75–98% gap closed	4
25–75% gap closed	6
<25% gap closed	4

of cuts is well know to practitioners, and has recently been demonstrated in [9, 22]. Another characteristic of split disjunctions, which is of practical interest, is the size of coefficients occurring in the support vector of the split disjunction. Note that cuts obtained from split disjunctions having very large coefficients

Table 5 Summary of coefficient size variation for pure IP instances (MIPLIB 3.0)

Instance	No. of integer-constrained variables	Mean coefficient size across all iterations
nw04	87,482	1.228
air05	7,195	1.156
seymour	1,372	1.099
misc03	159	1.227
p0033	33	2.099

Table 6 Summary of coefficient size variation for mixed IP instances (MIPLIB 3.0)

Instance	No. of integer-constrained variables	Mean coefficient size across all iterations
qnet1_o	1,417	2.381
gesa2_o	720	1.767
arki001	538	3.044
vpm1	168	1.833
pp08aCUTS	64	1.418

could be numerically unstable. Typically, split disjunctions with small coefficients are used in practice to generate split cuts. The best known upper bound on the coefficients of split disjunctions required to generate the split closure, however, is much larger (see [2,19]). Consequently, there is a huge gap between the bounds on the coefficients suggested by theory and those used in practice, and it is of interest to know whether a significant duality gap can be closed by using only split disjunctions with small coefficients. Interestingly, our experiment provides an affirmative answer to this question, at least on the test-bed of MIPLIB 3.0 instances: we found that most of the split disjunctions discovered during the separation phase did not have very large coefficients, the size of the problem notwithstanding. Table 5 shows the variation of the average coefficient size of the split disjunctions with the number of integer variables for five pure IP instances from MIPLIB 3.0. Similarly, Table 6 shows the variation of the average coefficient size of the split disjunctions with the number of integer variables for five mixed IP instances from MIPLIB 3.0.

These results suggest that most of the coefficients of the disjunctions used had values in $\{0, 1, -1\}$. An interesting question that arises is, whether restricting the coefficients of the disjunctions to the range $[-1, 1]$ would significantly affect the fraction of gap closed. While we do not have a complete answer to this question, our experiments on the structured problem classes indicate (see Sect. 4.2) that such a restriction can have a widely differing effect on the outcome, depending on the problem type: for some classes the effect is minimal, while for others it is substantial.

4.2 Structured problems

We also ran our code on some structured mixed integer programming instances discussed in the literature. For these instances we used the same stopping criteria as for the MIPLIB 3.0 instances, amended with a rule to terminate the run

for each instance after a time limit specific to each class (see Table 16). Full results on those instances are available at [26].

4.2.1 Location problems

Consider the following model for the capacitated warehouse location problem (CWLP).

$$
\begin{aligned}
\min \ & \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} + \sum_{i\in I} f_i y_i \\
\text{s.t.} \ & \\
& \sum_{i\in I} x_{ij} = 1 && \forall j \in J \\
& \sum_{j\in J} w_j x_{ij} \le s_i y_i && \forall i \in I \\
& x_{ij} \ge 0 && \forall i \in I, \forall j \in J \\
& y_i \in \{0,1\} && \forall i \in I.
\end{aligned}
$$

Here I is the set of warehouses, s_i is the capacity and f_i is the fixed cost associated with warehouse $i \in I$; while J is the set of customers, w_j is the demand of customer j and c_{ij} is the cost of serving customer j via warehouse i for all i, j. When the flow variables x_{ij} of the above model are also constrained to be 0-1, the resulting model is the formulation for the well-known single source capacitated facility location model. Similarly, if the sum of the y_i variables is bounded above by an integer p, the resulting model is the formulation for the capacitated p-median problem.

Operations Research Library (ORLIB, maintained by Beasley [13]) has two sets of CWLP instances. The first set has 37 instances, each one having 50 customers and 16–50 warehouses. The second set has 12 instances with 1,000 customers and 100 warehouses. We ran our code on all of the instances in each set. Table 7 summarizes our results on the first set of 37 instances. When a row refers to more than one instance, the numbers in each column define the range of the respective attribute. Note that the split closure closes 100% duality gap on all 37 instances in this set. Actually, strengthened rank-1 lift-and-project cuts from elementary disjunctions (i.e. split disjunctions on a single variable) are sufficient to solve the instances in the first set to optimality.

Table 8 summarizes our results on the second set of 12 CWLP instances. These instances are divided into types A, B and C differing in their c_{ij}, f_{ij} and w_j values; whereas within each type they differ only in the capacities s_i of the warehouses. On average, rank-1 split cuts close 94.54% duality gap on these instances.

We ran our code on 4 sets of benchmark problems for the single source capacitated facility location problem (SSCFLP), which were randomly generated by Holmberg et al. [24] (also see Ahuja et al. [1]). The test problems in each subset differ for the distributions of the inputs and for the size, which

Table 7 ORLIB capacitated warehouse location instances (Set 1)

Instance class	# Instances	# Warehouses	# Customers	% Gap closed	# Iterations by SC	# Cuts Total	# Cuts Binding for the last solution	Time (s)
cap41–cap44	4	16	50	100.00	16–85	61–439	34–79	9–324
cap51	1	16	50	100.00	25	185	107	33
cap61–cap64	4	16	50	100.00	6–31	51–229	39–120	1–64
cap71–cap74	4	16	50	100.00	7–8	52–72	42–64	1–3
cap81–cap84	4	25	50	100.00	8–61	88–462	74–135	12–2635
cap91–cap94	4	25	50	100.00	7–30	78–250	76–166	5–607
cap101–cap104	4	25	50	100.00	7–15	75–116	71–116	3–6
cap111–cap114	4	50	50	100.00	7–74	144–946	140–286	11–3123
cap121–cap124	4	50	50	100.00	7–37	175–439	170–321	36–380
cap131–cap134	4	50	50	100.00	9–22	166–321	160–301	1–17

Table 8 ORLIB Capacitated Warehouse Location Instances (Set 2) (1,000 customers, 100 warehouses)

Instance	Warehouse Capacity	% Gap closed by SC	# Iterations	# Cuts Total	# Cuts Binding for the last solution	Time (s)
capa_8000	8,000	88.65	110	7,040	2,906	36,926
capa_10000	10,000	86.30	91	5,940	3,190	36,685
capa_12000	12,000	92.77	89	5,495	3,031	36,542
capa_14000	14,000	99.00	32	2,669	2,546	12,881
capb_5000	5,000	95.04	72	4,286	2,669	36,397
capb_6000	6,000	95.03	74	4,765	3,006	36,938
capb_7000	7,000	95.49	75	5,073	3,375	36,419
capb_8000	8,000	97.22	71	4,810	3,326	26,793
capc_5000	5,000	95.18	79	4,621	2,725	36,256
capc_5750	5,750	95.77	76	4,786	2,965	38,078
capc_6500	6,500	96.59	69	4,561	2,962	36,330
capc_7250	7,250	97.49	69	4,735	3,171	36,456

Table 9 Holmberg single source capacitated facility location instances

Instance class	# Instances	# Facilities	# Customers	% Gap closed by SC	# Iterations	# Cuts Total	# Cuts Binding for the last solution	Time (s)
p1–p24	24	10–20	50	97.12–100.00	7–109	108–1,509	70–193	30–3842
p25–p40	16	30	150	86.84–100.00	101–249	2,279–3,779	239–551	9,710–18,272
p41–p55	15	10–30	70–100	96.41–100.00	17–247	231–3,686	84–416	96–18,144
p56–p71	16	30	200	88.43–100.00	48–92	1,464–3,038	319–604	9,016–18,573

ranges from 50 to 200 customers, and from 10 to 30 candidate facility locations. Table 9 summarizes our key findings, grouped by instance category. The split closure closes, on average, 98.56% of the duality gap on these instances.

Table 10 ORLIB capacitated p-median instances

Instance class	# Instances	# Facilities	# Customers	Median (P)	% Gap closed by SC	# Cuts		Binding for the last solution	Time (sec)
						# Iterations	Total		
p1–p10	10	50	50	5	99.82–100.00	19–126	544–2,783	312–534	104–6,183
p11–p20	10	100	100	10	99.53–100.00	60–90	2,069–3,403	780–952	3,466–10,727

We ran our code on a set of benchmark instances for the capacitated p-median problems, obtained from ORLIB repository [13]. Each one of these instances has the same number of customers and facilities. The first 10 instances (p1–p10) have 50 customers and 5 medians ($p = 5$), while the last 10 instances (p11–p20) have 100 customers and 10 medians ($p = 10$). Table 10 summarizes our key findings. The split closure closes, on average, around 99.92% of the duality gap on these instances.

4.2.2 Network design problems

We tested our code on two classes of network design problems. The first class consists of capacitated fixed charge network flow problems:

$$\begin{aligned} \min \quad & \sum_{a \in A} c_a x_a & & \\ \sum_{a \in A_i^+} y_a - \sum_{a \in A_i^-} y_a & \leq b_i & & \forall i \in N \\ y_a & \leq u_a x_a & & \forall a \in A \\ x_a & \in \{0,1\} & & \forall a \in A \\ y_a & \geq 0 & & \forall a \in A \end{aligned}$$

where N, A are the node set and arc set of the underlying network; A_i^+, A_i^- are the sets of arcs with heads and tails, respectively, at node $i \in N$, b_i is the supply at node i, u_a is the capacity, and c_a is the fixed cost of arc $a \in A$.

We ran our code on 2 sets of benchmark instances, which were randomly generated by Atamturk (see [3] for more details) and are available at [14]. Table 11 summarizes our key findings. The split closure closes, on average, 94.58% of the duality gap on these instances.

The second class consists of multicommodity capacitated network design problems with splittable and unsplittable flows [5]. Given a network, a set of commodities specified as origin-destination pairs, and demand data for the commodities, a multicommodity capacitated network design problem is to install integer multiples of some capacity unit on the arcs of the network and route the flow of commodities so that the total capacity installation and flow routing

Table 11 Fixed charge network flow instances

Instance Class	# Instances	# Nodes	Arc density (%)	% Gap closed by SC	# Iterations	# Cuts		Time (s)
						Total	Binding for the last solution	
fc.30.50	10	30	50	90.63–99.02	93–203	4,894–8,859	132–202	5,079–14,993
fc.60.20	10	60	20	84.48–97.72	95–149	6,845–9,350	212–295	15,985–18,186

Table 12 Multi-commodity capacitated network design instances (splittable version)

Instance class	# Instances	# Constraints	# Variables		% Gap closed by SC	# Iterations	# Cuts		Time (sec)
			Total	Integer			Total	Binding for the last solution	
ns4_pr	5	1,138–2,639	3,393–8,601	29–61	76.72–93.37	365–2,302	7,928–41,956	109–244	65,784–86,845
ns25_pr	5	1,138–2,639	3,393–8,601	29–61	55.83–86.33	186–3,091	6,333–69,388	194–276	69,889–87,109
ns60_pr	5	1,138–2,639	3,393–8,601	29–61	65.58–93.69	176–1,752	8,905–44,411	310–546	86,400–87,072

costs are minimized, while meeting the demands for the commodities. A typical integer programming model for these problems has the following constraints for each arc of the network:

$$\sum_{k \in K} d_k x_{ka} \leq c_{a0} + c_a y_a$$
$$y_a \in \mathbb{Z}_+$$
$$0 \leq x_{ka} \leq 1\,,$$

where K denotes the set of commodities, d_k is the demand for the commodity k, c_{a0} is the existing capacity of the arc and c_a is the unit capacity some integer multiple of which is to be installed; x_{ka} are the flow variables which might be continuous (*splittable*) or discrete (*unsplittable*) depending on whether the flow of commodities is allowed to be split among several paths.

We ran our code on a set of benchmark instances for the splittable and unsplittable flow capacitated network design problem, which are described in Atamturk and Rajan [5] and are available at [14]. Table 13 summarizes our key findings. The split closure closes, on average, 78.24 and 67.40% of the duality gap on splittable and unsplittable versions of these problems, respectively.

4.2.3 Lot sizing problems

Consider the following capacitated version of the n-period lot-sizing problem.

$$\min \sum_{t=1}^{n} (h_t i_t + p_t w_t + s_t z_t)$$

Table 13 Multi-commodity capacitated network design instances (unsplittable version)

Instance class	# Instances	# Constraints	# Integer Variables	% Gap closed by SC	# Iterations	# Cuts		Time (sec)
						Total	Binding for the last solution	
nu4_pr	5	1,138–2,639	3,393–8,601	50.75–94.52	16–103	392–3,793	26–288	678–52,323
nu25_pr	5	1,138–2,639	3,393–8,601	44.33–81.92	49–301	2,259–26,381	139–418	5,906–87,368
nu60_pr	5	1,138–2,639	3,393–8,601	46.52–72.23	39–223	3,322–13,452	235–429	86,486–87,095
nu120_pr	5	1,138–2,639	3,393–8,601	43.84–88.55	35–247	3,389–9,921	236–456	86,651–89,947

$$
\begin{aligned}
\text{s.t}\quad & \\
& i_t + w_t - i_{t+1} = d_t \quad \forall t \in \{1,\ldots,n\} \\
& w_t \le c_t z_t \quad t \in \{1,\ldots,n\} \\
& i_{n+1} = 0 \\
& w_t \ge 0 \quad t \in \{1,\ldots,n\} \\
& i_t \ge 0 \quad t \in \{1,\ldots,n+1\} \\
& z_t \in \{0,1\} \quad t \in \{1,\ldots,n\}\,.
\end{aligned}
$$

Here p_t, h_t and s_t denote the production, holding and setup costs in period t; w_t, i_t and z_t denote the production, incoming inventory and setup variables in period t; d_t is the demand and c_t is the production capacity in period $t \in \{1,\ldots,n\}$.

We ran our code on a set of benchmark instances for the capacitated lot sizing model, which was created by Atamturk et al. [4] and is available at [14]. All instances in this data set are 90-period problems ($n = 90$) with varying cost and capacity characteristics (see [4] for details). Table 14 describes our key findings. The split closure closes, on average, 79.37% of the duality gap on these instances.

Table 14 Capacitated lot sizing instances

Instance class	# Instances	% Gap closed by SC	# Iterations	# Cuts		Time (sec)
				Total	Binding for the last solution	
cls.T90.C2.	25	72.61–94.64	18–138	311–2,957	41–81	1,298–12,758
cls.T90.C3.	25	69.46–89.09	11–111	377–1,751	52–97	1,020–9,447
cls.T90.C4.	25	72.72–90.16	20–142	368–2,884	67–102	1,258–18,135
cls.T90.C5.	25	71.37–95.42	18–133	393–2,203	65–118	867–10,524

Table 15 Problem statistics (arki001)

	Original	Preprocessed
Rows	1,048	782
Columns	1,388	959
General integer variables	123	96
Binary variables	415	387
Continuous variables	850	476

4.3 An unsolved instance solved

Note that rank-1 split cuts can be used to strengthen the formulation of a MIP problem before it is fed to a general purpose MIP solver, as observed by Fischetti and Lodi [23] in the context of rank-1 Chvatal-Gomory cuts. We next apply this paradigm to solve to optimality the very hard instance arki001 from MIPLIB 3.0. This instance (also present in MIPLIB 2003) is one of the very few unsolved ones in the MIPLIB 3.0 test-bed. This problem originates in the metallurgical industry. Its characteristics are described in Table 15.

We preprocessed the instance using the CPLEX 9.0 presolver. The formulation of the preprocessed problem was then strengthened by generating rank-1 split cuts. We were able to close 83.05% of the duality gap by using only rank-1 split cuts. The rank-1 separation procedure was terminated after the code failed to produce a valid separating rank-1 split cut in an hour. The strengthened formulation containing only those split cuts which were binding at the optimum of the MP was stored in a file, and given to the CPLEX 9.0 MIP solver. The MIP solver was configured to use strong branching as the variable selection strategy, and was run in the "emphasize optimality" mode. CPLEX 9.0 was able to solve the resulting formulation to optimality in about 11 hours. Table 15 shows the statistics associated with this problem solving exercise.

For the sake of comparison, we also ran CPLEX 9.0 on the preprocessed instance without the split cuts for a period of 100 h (360,000 s). Figure 2 displays the growth of the percentage duality gap closed with respect to time in the two cases. The thin line represents the run of CPLEX without split cuts, while the thick line traces the run of CPLEX aided by the rank-1 split cuts. Note that without the cuts, CPLEX 9.0 was unable to close more than 84.11% of the duality gap after enumerating about 43 million branch and bound nodes (22 million active nodes) in 100 h of CPU time. The sudden jump at 200,000 s represents the crossover point of the experiment, i.e the time at which we terminated the generation of rank-1 split cuts and gave the strengthened formulation to CPLEX 9.0.

It is interesting to compare the information available at the crossover point (at 200,000 s) in each of the above two setups. CPLEX, acting on the unstrengthened formulation, was able to close around 82.67% of the duality gap (in addition to discovering integer feasible solutions) after enumerating around 24 million branch and bound nodes in 200,000 s. In the remaining 160,000 s, it closed an additional 1.44% of the gap. Our rank-1 separation procedure

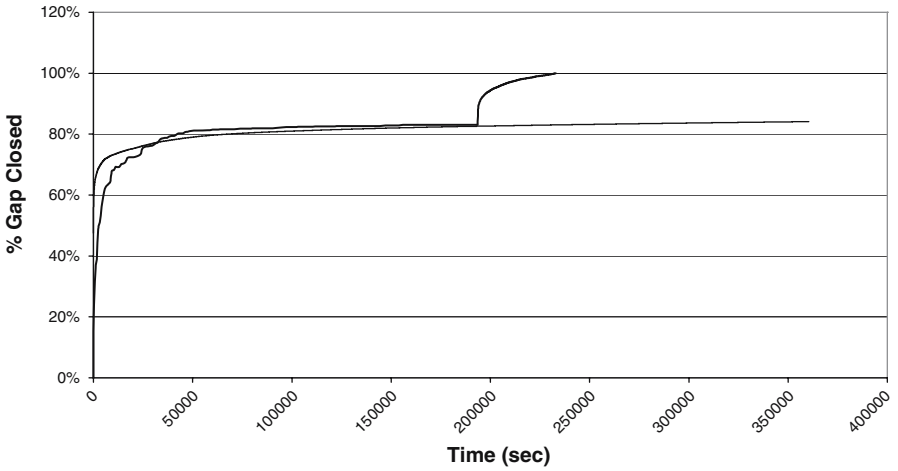

Fig. 2 Percentage duality gap closed w.r.t. time: CPLEX unaided (thin line) versus CPLEX applied after adding rank 1 split cuts (thick line)

closed 83.05% of the gap in 200,000 s. While both procedures closed about the same proportion of the gap in the first 200,000 s, the *representation* of this information is radically different in the two cases. At the crossover point, the rank-1 strengthening procedure had the lower bound information represented in the *aggregated* form of 227 rank-1 split cuts. CPLEX, on the other hand, had roughly the same lower bound information in a *disaggregated* form – in the form of a huge branch-and-bound tree, containing around 24 million nodes. It is this difference in information representation, which finally made it possible to solve arki001 to optimality.

According to the latest addition to the website of MIPLIP 3.0, R. Bixby announced in June 2006 the solution of arki001 with CPLEX 10.0 after running it for more than a month and generating more than 100 million nodes.

5 Concluding remarks

In this project we set out to explore the strength of the elementary split closure, i.e. the set defined by all rank 1 split cuts. We used an iterative procedure to optimize over the split closure, whose emphasis was on getting as close as possible to the optimum, notwithstanding the computational cost. Our primary finding is that this optimum closes in most cases a remarkably high fraction of the integrality gap (72% on average). Furthermore, we found that generating rank 1 split cuts is a remarkably stable procedure: our code ran on some of the instances for several days at a time, generating tens of thousands of cuts without numerical difficulties, and with the last cuts generated being as clean as the first ones.

We also found that although the split closure is computationally expensive to generate, the collection of cuts binding at the last iteration can be used to tighten

Table 16 Summary for structured instances

Problem class	Source	# Instances	% Gap closed by SC	% Gap closed by unstrengthened cuts from			Time Limit (s)
				$\{-1, 0, 1\}$ Split Disjunctions (%)	$\{0, 1\}$ Split Disjunctions (%)	Elementary Split Disjunctions (%)	
Capacitated warehouse location (Set 1)	ORLIB [13]	37	100.00	99.95	99.94	98.32	36,000
Capacitated warehouse location (Set 2)	ORLIB [13]	12	94.54	94.52	94.51	94.33	36,000
Capacitated p-Median	ORLIB [13]	20	99.92	99.91	99.91	98.47	18,000
Single source capacitated facility location	Holmberg [24]	71	98.56	94.51	92.73	88.73	18,000
Fixed charge network flow	BCOL [14]	20	94.58	31.36	29.44	2.35	18,000
Multi-commodity Capacitated network design (splittable version)	BCOL [14]	15	78.24	77.51	69.97	46.16	86,400
Multi-commodity capacitated network design (Unsplittable Version)	BCOL [14]	20	67.40	38.98	20.41	15.92	86,400
Capacitated lot sizing	BCOL [14]	100	79.37	78.49	78.48	26.46	18,000

the original problem formulation to an extent that sometimes makes a previously unsolvable problem solvable (e.g. arki001). This opens up the possibility of a practical use of our procedure, to solve very hard integer programming instances. Furthermore, our algorithm can be run in parallel with a generic MIP solver, without interfering with the latter, other than by periodically adding cuts to its pool.

As a byproduct of our experiment, we found not only that the average coefficient size of the split disjunctions used is very small, but more specifically, that restricting the use of cuts to unstrengthened cuts from disjunctions with coefficients equal to 0, 1 or -1, for many problem classes (especially structured ones) does not affect substantially the gap closed (see Table 16).

References

1. Ahuja, R.K., Orlin, J.B., Pallottino, S., Scaparra, M.P., Scutellà, M.G.: A multi-exchange heuristic for the single source capacitated facility location problem. Manage. Sci. **50**(6), 749–760 (2004)
2. Andersen, K., Cornuéjols, G., Li, Y.: Split closure and intersection cuts. Math. Program. A **102**, 457–493 (2005)
3. Atamtürk, A.: Flow pack facets of the single node fixed-charge flow polytope. Oper. Res. Lett. **29**, 107–114 (2001)
4. Atamtürk, A., Munóz, J.C.: A study of the lot-sizing polytope. Math. Prog. **99**, 443–465 (2004)
5. Atamtürk, A., Rajan, D.: On splittable and unsplittable capacitated network design arc-set polyhedra. Math. Program. **92**, 315–333 (2002)
6. Balas, E.: Disjunctive programming. Ann. Discrete Math. **5**, 3–51 (1979)
7. Balas, E., Ceria, S., Cornuéjols, G.: A lift-and-project cutting plane algorithm for mixed 0–1 programs. Math. Program. **58**, 295–324 (1993)
8. Balas, E., Ceria, S., Cornuéjols, G.: Mixed 0-1 programming by lift-and-project in a branch-and-cut framework. Manage. Sci. **42**, 1229–1246 (1996)
9. Balas, E., de Souza, C.: The vertex separator problem: a polyhedral investigation. Math. Program. **103**(3), 583–608 (2005)
10. Balas, E., Perregaard, M.: Lift and project for mixed 0–1 programming: recent progress. Discrete Appl. Math. **123**, 129–154 (2002)
11. Balas, E., Perregaard, M.: A precise correspondence between lift-and-project cuts, simple disjunctive cuts, and mixed integer Gomory cuts for 0–1 programming. Math. Program. B **94**, 221–245 (2003)
12. Balas, E., Saxena, A.: Separation functions in disjunctive programming (in preparation)
13. Beasley, J.E.: OR-Library, people.brunel.ac.uk/∼mastjjb/jeb/info.html
14. Berkeley Computational Optimization Lab, http://ieor.berkeley.edu/∼atamturk/data
15. Bonami, B., Minoux, M.: Using rank 1 lift-and-project closures to generate cuts for 0-1 MIP's, a computational investigation. Discrete optim. **2**, 288–308 (2005)
16. Bonami, P., Cornuéjols, G., Dash, S., Fischetti, M., Lodi, A.: Projected Chvátal-Gomory cuts for mixed integer linear programs. Math. Program. A (to appear)
17. Caprara, A., Letchford, A.: On the separation of split cuts and related inequalities. Math. Program. **94**, 279–294 (2003)
18. COIN: Computational infrastructure for operations research, http://www.coin-or.org
19. Cook, W.J., Kannan, R., Schrijver, A.: Chvatal closures for mixed integer programming problems. Math. Program. **47**, 155–174 (1990)
20. Cornuéjols, G., Li, Y.: Elementary closures for integer programs. Oper. Res. Lett. **28**, 1–8 (2001)
21. Dash, S., Günlük, O., Lodi, A.: Optimizing over the MIR closure. Talk presented at INFORMS Meeting (San Francisco), November (2005)
22. de Souza, C., Balas, E.: The vertex separator problem: algorithms and computations. Math. Program. **103**(3), 609–631 (2005)
23. Fischetti, M., Lodi, A.: Optimizing over the first Chvátal closure. Math. Program. B (to appear)

24. Holmberg, K., Ronnqvist, M., Yuan, D.: An exact algorithm for the capacited facility location problems with single sourcing. Eur. J. Oper. Res. **113**, 544–559 (1999)
25. Nazareth, J.L.: The homotopy principle and algorithms for linear programming. SIAM J. Optim. **1**, 316–332 (1991)
26. Saxena, A.: OSCLIB, http://www.andrew.cmu.edu/user/anureets/osc/osc.htm