Carnegie Mellon University
Research Showcase @ CMU

Tepper School of Business

12-2005

Optimizing over the Split Closure

Egon Balas
Carnegie Mellon University, eb17@andrew.cmu.edu

Anureet Saxena
Carnegie Mellon University

Follow this and additional works at: http://repository.cmu.edu/tepper

Part of the Economic Policy Commons, and the Industrial Organization Commons

Recommended Citation
Mathematical Programming, 113, 2, 219- 240.

This Article is brought to you for free and open access by Research Showcase @ CMU. It has been accepted for inclusion in Tepper School of Business by an authorized administrator of Research Showcase @ CMU. For more information, please contact research-showcase@andrew.cmu.edu.

Optimizing over the Split Closure

Egon Balas Anureet Saxena

December 2005

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 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. We show that this separation problem can be formulated 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 Capacitated Warehouse Location Problems from OrLib.

Our computational results show that rank-1 split cuts close more than 98% of the duality gap on 14 out of 33 mixed integer instances from MIPLIB 3.0. More than 75% of the duality gap can be closed on an additional 11 instances. The average gap closed over all 33 instances is 82%. In the pure integer case, rank-1 split cuts close more than 75% of the duality gap on 13 out of 25 instances from MIPLIB 3.0. On average, rank 1 split cuts close about 71% of the duality gap on these 25 instances. We also report our results on two sets of real-world capacitated warehouse location problem (CWLP) instances from OrLib. The first set has 37 instances, each one having 50 customers and 16-50 warehouses. The second set has 12 instances with 1000 customers and 100 warehouses. The split closure closes 100% of the duality gap on all the 37 instances from the first set, and on average about 93% of the duality gap on the 12 instances from the second set.

We also gathered statistics on the support size of the disjunctions generated (which affects the density of the resulting cut) and the average size (absolute value) of their coefficients. 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 (MIP) be

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

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*

$$\pi x \leq \pi_0 \quad \vee \quad \pi x \geq \pi_0 + 1$$

is obviously satisfied by any $x \in P_I$. 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 [2, 3]), where

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

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 [13]) are a special class of disjunctive cuts [2] known in the more recent literature as lift-and-project cuts [3]. For the connection between split cuts, lift-and-project cuts and mixed integer Gomory cuts, see [7, 14, 11].

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 [17] have pointed out a way to do this for the case of the elementary Chvatal closure of a pure integer program. Work in this vein is also reported in [10, 12, 15]. In this paper we take a similar approach to optimizing over the split closure of a mixed integer program.

Given a mixed integer program of the form (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

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

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 x^*, the Separation Problem is of the form

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

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 x^* (if $\alpha x^* - \beta < 0$) or else proves that $x^* \in \mathcal{C}$.

Since $\mathcal{C}$ is known to be a polyhedron [13], this procedure ends in a finite number of iterations by finding the minimum of cx over $\mathcal{C}$. 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, which cannot be linearized to the best of our knowledge. 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, imposing the condition $u_0 > 0$, $v_0 > 0$ known to be satisfied by every solution that yields a nontrivial cut (Lemma 1 of [6]), 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

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

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.

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

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

where $\hat{s} := A\hat{x} - b$.

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 $\alpha\bar{x} - \beta$. $\square$

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.

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

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 [18]).

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. $\square$

The problem (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*

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

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 $\bar{z}(\bar{\theta}) = \hat{z}(1 - \bar{\theta})$. $\square$

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).

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, Ceria and Cornuéjols [4] 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 optimal LP solution, plus the nonbasic slacks, and lift the resulting cuts to the full space as in [4].

While (PMILP) could 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, doing this would prevent us from using a state-of-the-art code 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 $\{0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$, 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

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

used in [6, 7]. 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), where we set $k = 2|e| + 2$. 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), 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 cuts by enforcing their partial orthogonality in some directions. To be specific, every time we move to a new grid point, we impose the condition $\pi_j = 0$, $j \in N_1^*$, on a heuristically chosen subset N_1^* of variables in order to diversify the set of disjunctions generated.

A flow chart of our algorithm is shown in Figure 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, the following result throws some light on the accuracy of our bound.

For given (π, π_0), let

$$\rho := \max\{\pi x : Ax \geq b\} - \min\{\pi x : Ax \geq b\}$$

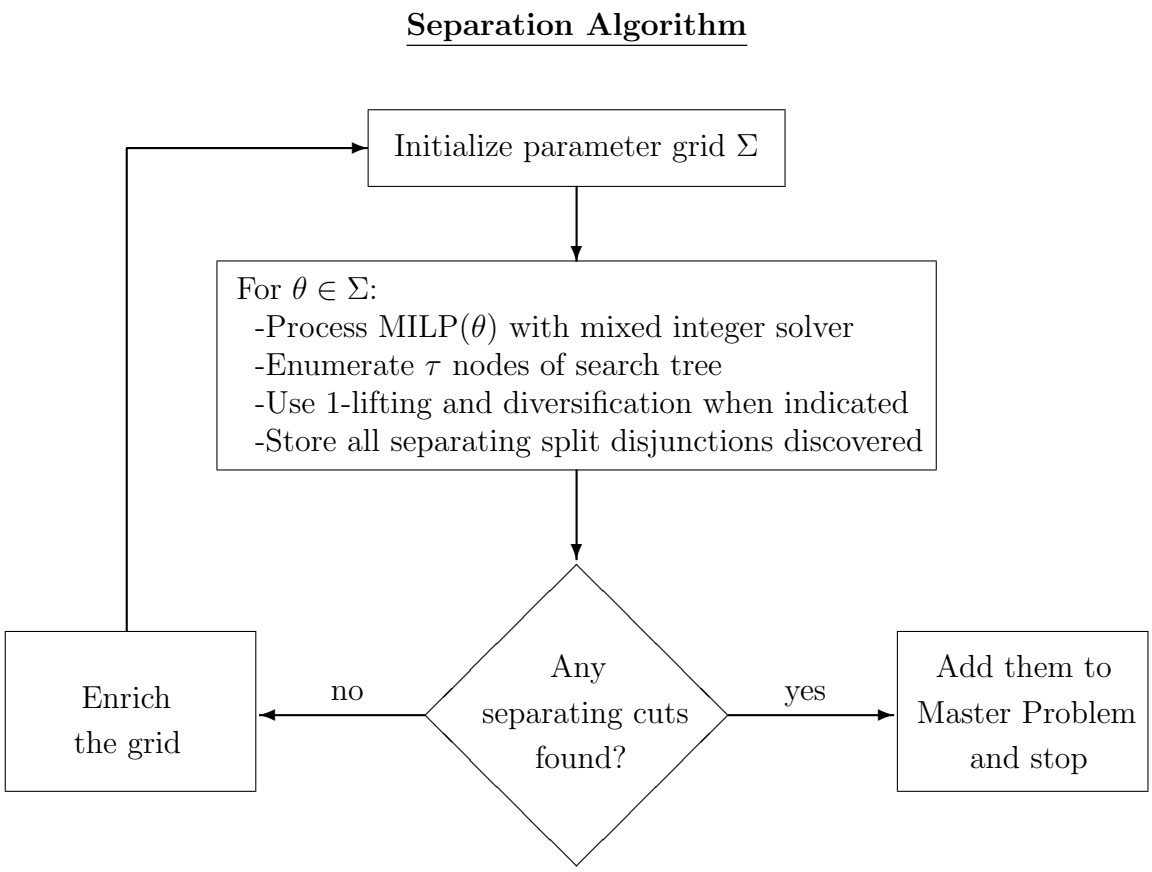

Figure 1: Flowchart of Separation Algorithm

be the *width* of the disjunction defined by (π, π_0). Note that ρ increases with $|\pi_j|$, $j \in N$. Further, let $-\eta$ be the optimum of (SP) over all values of θ, $0 \leq \theta \leq 1$, for (π, π_0). Finally, let d be the distance between the grid points. Then it can be shown [8] that

$$\rho \geq \eta(\frac{1}{d} - 1) + 1.$$

In other words, the width of the disjunction defined by (π, π_o), hence the size of the coefficients $|\pi_j|$, increases with the density $(\frac{1}{d})$ of the grid.

Consequently, for a sufficiently dense grid, a split disjunction with very large coefficients would be required to cut off the last incumbent solution. But as our computational experiments show (see the next section, tables 7-8), the likelihood of the occurrence of such a disjunction is insignificant.

4 Computational Results

We implemented the separation algorithm (discussed in the previous section) using COIN-OR [12] 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 MIPLIB 3.0 instances and OrLib Capacitated Warehouse Location problem instances.

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 was unable to find a new violated cut within 1 hour from the last one. The percentage of duality gap closed, computed as 100-100×(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* program (MIP) instances from MIPLIB 3.0. The second column of the table gives the total number of variables. The third column gives the number of integer constrained variables, whereas the fourth column gives the number of integer variables which are 0-1 constrained. 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. Finally, 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 [10]. Table 2 summarizes these results.

Table 3 reports our results on all of the *pure integer* programs from MILIB 3.0. All of these instances except for gt2, have only binary variables. The second column of the table gives the number of (integer constrained) variables. The third column gives the best lower-bound obtainable by our code on the gap closed by the elementary split closure. Finally, for the sake of comparison, the fourth column gives the best lower bound obtained by the Fischetti-Lodi code [17] on the gap closed by the first Chvatal closure. Table 4 summarizes these results. The averages quoted in Table 4 are obtained after excluding the instances enigma (duality gap = 0) and harp2 (numerically unstable).

The *support* of a split disjunction, $D(\pi, \pi_0)$, is the set of non-zero components of π. We are considering density (size of support) of a disjunction rather than that of the cut that it gives rise to, because a given disjunction can give rise to different cuts, depending on the basis used in deriving them. Split disjunctions with sparse support tend to give rise to sparse split cuts. Sparse cuts are naturally preferred to dense cuts in practical cutting plane algorithms, as they avoid slowing down the computations and possibly leading to numerical difficulties. The importance of sparsity has been documented in some recent work [5, 16].

In our experiment, we found that most of the split disjunctions discovered during the separation phase had support of 10-20 columns, the size of the problem notwithstanding. Figures 2-6 show the variation of the average support size of the split disjunctions with the

	# Variables					
Instance	Total	General Integer	Binary	% Gap SC	% Gap pro-CG	Comments
10teams	2025	1800	ALL	100.00	57.1400	
arki001	1388	538	415	83.05	28.0400	p
bell3a	133	71	39	55.19	48.1000	a, p
bell5	104	58	30	87.44	91.7300	a
blend2	353	264	231	46.77	36.4000	
dano3mip	13873	552	ALL	0.12	0.0000	
danoint	521	56	ALL	7.44	0.0100	
dcmulti	548	75	ALL	100.00	47.2500	
dsbmip	1886	192	160	–	0.0000	z
egout	141	55	ALL	100.00	81.7700	a
fixnet6	878	378	ALL	99.76	67.5100	a
flugpl	18	11	0	100.00	19.1900	
gen	870	150	144	100.00	86.6000	
gesa2	1224	408	240	98.66	94.8400	
gesa2_o	1224	720	336	100.00	94.9300	
gesa3	1152	384	216	95.78	58.9600	
gesa3_o	1152	672	336	95.31	64.5300	
khb05250	1350	24	ALL	100.00	4.7000	a
misc06	1808	112	ALL	100.00	0.0000	a
mod011	10958	96	ALL	80.72	0.0000	a, p
modglob	422	98	ALL	96.48	0.0000	a, p
noswot	128	100	75	–	0.0000	z
pk1	86	55	ALL	0.00	0.0000	a
pp08aCUTS	240	64	ALL	97.01	0.6800	
pp08a	240	64	ALL	95.81	4.3200	
qiu	840	48	ALL	77.51	10.7100	
qnet1	1541	1417	1288	100.00	7.3200	
qnet1_o	1541	1417	1288	100.00	8.6100	
rgn	180	100	ALL	100.00	0.0000	a
rout	556	315	300	70.73	0.0300	
set1ch	712	240	ALL	89.41	51.4100	a
vpm1	378	168	ALL	100.00	100.0000	a
vpm2	378	168	ALL	81.22	62.8600	a

^{p} the original formulation was preprocessed using the CPLEX 9.0 presolver; the percentage gap reported is w.r.t the preprocessed problem.

^{z} the initial duality gap is 0.00, i.e value of the LP relaxation is equal to the value of the IP optimum.

^{a} the gap reported under the pro-CG column is defined by the actual optimum over the first pro-CG closure

Table 1: Mixed IP instances (MIPLIB 3.0)

Total Number of Instances	33
Number of Mixed 0/1 Instances	18
Average %Gap closed by Split Closure*	82.53%
98-100% Gap closed	14 instances
75-98% Gap closed	11 instances
25-75% Gap closed	3 instances
$< 25\%$ Gap closed	5 instances

*average taken over instances for which there exists a positive duality gap.

Table 2: Summary of mixed IP instances (MIPLIB 3.0)

number of iterations for five pure IP instances from MIPLIB 3.0. Table 5 summarizes the results for these instances. Similarly, figures 7-11 show the variation of the average support size of the split disjunctions with the number of iterations for five mixed IP instances from MIPLIB 3.0. Table 6 summarizes the results for these instances.

Another characteristic of split disjunctions which is of practical interest is the magnitude of coefficients occurring in the support vector of the split disjunction. Note that split cuts obtained from split disjunctions having very large coefficients could be numerical 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 Andersen, Cornuejols and Li [1]). Consequently, there is a huge gap between the bounds on the coefficients suggested by theory (see [1]) 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. Figures 12-16 show the variation of the average coefficient size of the split disjunctions with the number of iterations for five pure IP instances from MIPLIB 3.0. Table 7 summarizes the results for these instances. Similarly, figures 17-21 show the variation of the average coefficient size of

Instance	# Variables	% Gap Closed (Split Closure)	%Gap Closed (Chvatal Closure)	Comments
air03	10757	100.00	100.00	a
air04	8904	62.42	27.60	p
air05	7195	62.05	15.50	
cap6000	6000	37.63	26.90	p
enigma	100	–	–	z
fast0507	63009	18.08	4.70	
fiber	1254	98.50	98.50	
gt2	188	100.00	100.00	a
harp2	2993	17.50	29.00	n
l152lav	1989	92.10	69.20	
lseu	89	93.75	91.30	a
misc03	159	51.47	51.20	
misc07	259	19.48	16.10	
mitre	10724	100.00	100.00	a
mod008	319	100.00	100.00	a
mod010	2655	100.00	100.00	a
nw04	87482	100.00	100.00	a
p0033	33	87.42	85.40	a
p0201	201	74.93	60.50	
p0282	282	99.90	99.90	
p0548	548	100.00	100.00	a
p2756	2756	92.32	69.20	
seymour	1372	61.94	23.50	
stein27	27	0.00	0.00	a
stein45	45	0.00	0.00	

^{p}the original formulation was preprocessed using the CPLEX 9.0 presolver. The percentage gap reported is w.r.t the preprocessed problem.

^{z}the initial duality gap is 0.00, i.e value of the LP relaxation is equal to the value of the IP optimum.

^{a}the gap reported under the Chvatal Closure column is defined by the actual optimum over the first Chvatal closure.

^{n}the code encountered numerical problems

Table 3: Pure IP instances (MIPLIB 3.0)

Number of Instances	25
Average Gap Closed by the Split Closure	71.71%
98-100% Gap closed	9 instances
75-98% Gap closed	4 instances
25-75% Gap closed	6 instances
$<$ 25% Gap closed	6 instances
Average Gap Closed by the CG Closure	62.59%

Table 4: Summary of pure IP instances (MIPLIB 3.0)

Instance	No. of Integer-Constrained Variables	Mean Support Size across all Iterations
nw04	87482	2.084
air05	7195	8.210
seymour	1372	5.263
misc03	159	3.771
p0033	33	4.847

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

Instance	No. of Integer-Constrained Variables	Mean Support Size across all Iterations
qnet1_o	1417	6.690
gesa2_o	720	4.937
arki001	538	3.146
vpm1	168	4.503
pp08aCUTS	64	3.850

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

the split disjunctions with the number of iterations for five mixed IP instances from MIPLIB 3.0. Table 8 summarizes the results for these instances.

Instance	No. of Integer-Constrained Variables	Mean Coefficient Size across all Iterations
nw04	87482	1.228
air05	7195	1.156
seymour	1372	1.099
misc03	159	1.227
p0033	33	2.099

Table 7: Summary of coefficient size variation for pure IP instances (MIPLIB 3.0)

Instance	No. of Integer-Constrained Variables	Mean Coefficient Size across all Iterations
qnet1_o	1417	2.381
gesa2_o	720	1.767
arki001	538	3.044
vpm1	168	1.833
pp08aCUTS	64	1.418

Table 8: Summary of coefficient size variation for mixed IP instances (MIPLIB 3.0)

4.2 Instance arki001 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 [17] 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. The origin of this problem can be traced to metallurgical industry. Its characteristics are described in Table 9.

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

	Original	Preprocessed
Rows	1048	782
Columns	1388	959
General Integers Variables	123	96
Binary Variables	415	387
Continuous Variables	850	476

Table 9: Problem statistics (arki001)

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 master problem (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 while branching and was used in the "emphasize optimality" mode. CPLEX 9.0 was able to solve the resulting formulation to optimality in about 11 hours. Table 10 shows the statistics associated with this problem solving exercise.

Optimal solution value	7580813.046
LP relaxation value	7579599.808
% Gap closed by rank-1 split cuts	83.05%
Time spent in generating rank-1 split cuts	53.76 hrs
Time taken by CPLEX 9.0 after strengthening	10.94 hrs
No. of branch-and-bound nodes enumerated by CPLEX	643425
Total time taken to solve the instance to optimality	64.70 hrs

Table 10: Solution procedure statistics for arki001

Figure 22 displays the growth of the percentage duality gap closed with respect to time. The sudden jump at 200,000 seconds represent 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.

For the sake of comparison, we also ran CPLEX 9.0 on the preprocessed instance without the split cuts for a period of 100 hours (360,000 seconds). Figure 23 displays the growth of the percentage duality gap closed by CPLEX without 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 hours of CPU time.

It is interesting to compare the information available at the crossover point (at 200,000 sec) 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 seconds. In the remaining 160,000 seconds, it closed an additional 1.44% of the gap. Our rank-1 separation procedure closed 83.05% of the gap in 200,000 seconds. While both procedures closed about the same proportion of the gap in the first 200,000 seconds, 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.

4.3 OrLib Capacitated Warehouse Location Problems

Capacitated warehouse location problem (CWLP) can formulated as,

$$\begin{array}{rcll} \min \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} + \sum_{i\in I} f_i y_i & & & \\ \sum_{i\in I} x_{ij} & = & 1 & \forall j \in J \\ \sum_{j\in J} w_j x_{ij} & \leq & s_i y_i & \forall i \in I \\ x_{ij} & \geq & 0 & \forall i \in I, j \in J \\ y_i & \in & \{0,1\} & \forall i \in I \end{array}$$

where, I is the set of warehouses, s_i is the capacity and f_i is fixed cost associated with warehouse i for $i \in I$; 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.

Operations Research Library (OrLib, maintained by J. Beasley [9]) has two sets of CWLP instances. The first set has 37 instances, each one having 50 customers and 16 to 50 warehouses. The second set has 12 instances with 1000 customers and 100 warehouses. We ran our code on all of the instances in each set. For each instance, the experiment was terminated when an integer optimum solution was discovered or a pre-determined time-limit of 3 hours was violated. The percentage of duality gap closed, computed as 100-100×(remaining gap/original gap), is an underestimate as we only have a lower bound on the optimal value over the split closure.

Table 10 summarizes our results on the first set of 37 instances. The second column of the table gives the number of warehouses in each collection of problems and the third column gives the percentage duality gap closed by split closure. All the instances in this set have 50 customers. 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 0-1 disjunctions are sufficient to solve the instances in the first set to optimality.

Instances	#Warehouses	% Gap SC
cap41, cap42, cap43, cap44	16	100.00%
cap51	16	100.00%
cap61, cap62, cap63, cap64	16	100.00%
cap71, cap72, cap73, cap74	16	100.00%
cap81, cap82, cap83, cap84	25	100.00%
cap91, cap92, cap93, cap94	25	100.00%
cap101, cap102, cap103, cap104	25	100.00%
cap111, cap112, cap113, cap114	50	100.00%
cap121, cap122, cap123, cap124	50	100.00%
cap121, cap122, cap123, cap124	50	100.00%

Table 11: OrLib CWLP set 1 instances

Table 11 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. The first column of the table gives the name of the instance, the second column gives the common capacity of the warehouses and the last column gives the percentage duality gap closed by our code within 3 hours. On average, rank-1 split cuts close around 92.93% duality gap on these instances.

Instance	Type	Warehouse Capacity	% Gap SC
capa_8000	A	8000	88.38%
capa_10000	A	10000	87.57%
capa_12000	A	12000	91.53%
capa_14000	A	14000	97.53%
capb_5000	B	5000	94.18%
capb_6000	B	6000	92.80%
capb_7000	B	7000	92.42%
capb_8000	B	8000	93.74%
capc_5000	C	5000	93.42%
capc_5750	C	5750	93.90%
capc_6500	C	6500	94.69%
capc_7250	C	7250	95.06%

Table 12: OrLib CWLP set 2 instances

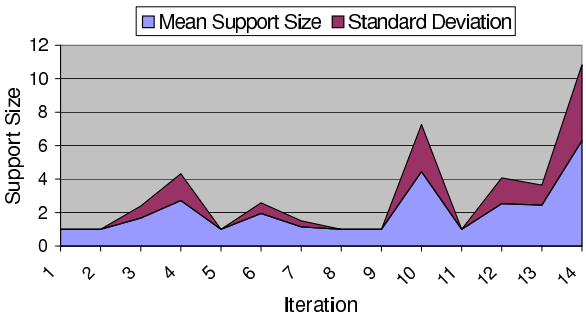

Figure 2: nw04: 87482 integer variables, 100% gap closed by split closure

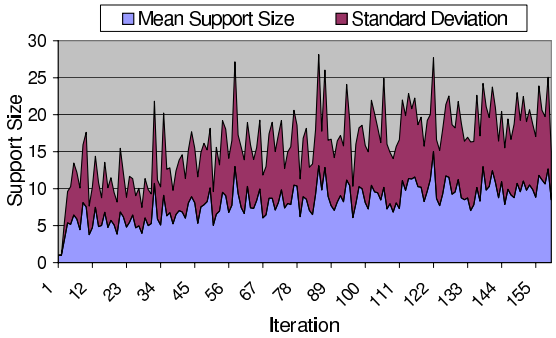

Figure 3: air05: 7195 integer variables, 62.05% gap closed by split closure

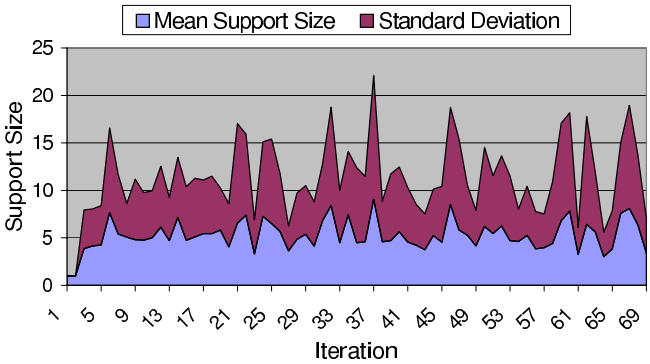

Figure 4: seymour: 1372 integer variables, 61.94% gap closed by split closure

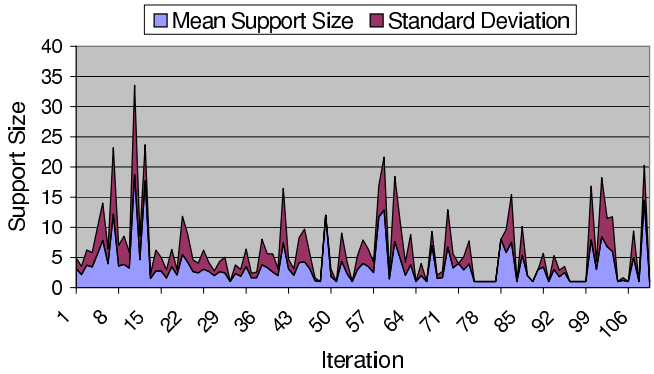

Figure 5: misc03: 159 integer variables, 51.47% gap closed by split closure

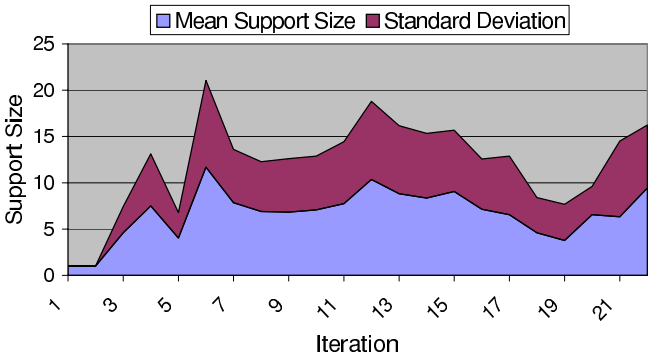

Figure 6: p0033: 33 integer variables, 87.42% gap closed by split closure

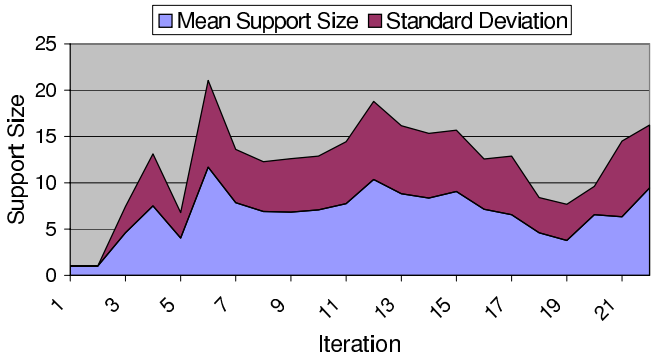

Figure 7: qnet1_o: 1417 integer variables, 100% gap closed by split closure

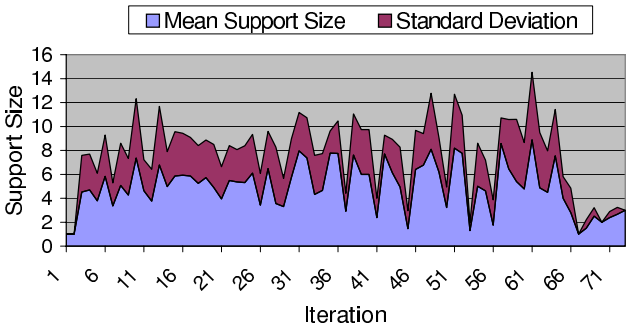

Figure 8: gesa2_o: 720 integer variables, 100.00% gap closed by split closure

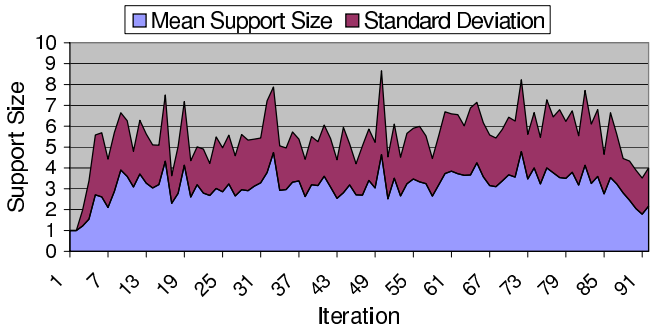

Figure 9: arki001: 538 integer variables, 83.05% gap closed by split closure

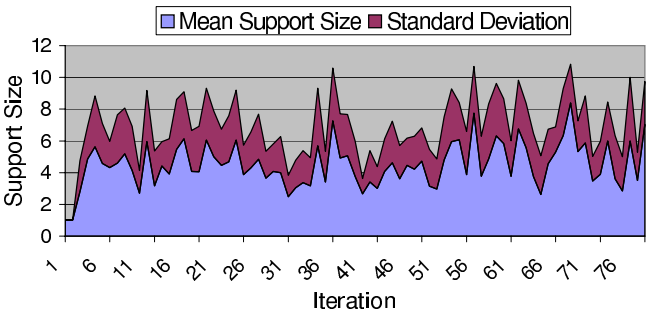

Figure 10: vpm1: 168 integer variables, 100% gap closed by split closure

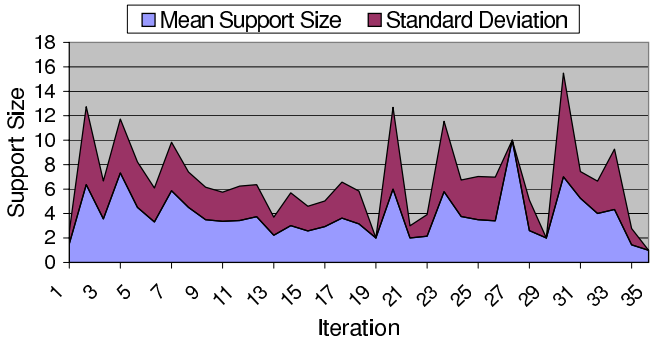

Figure 11: pp08aCUTS: 64 integer variables, 97.01% gap closed by split closure

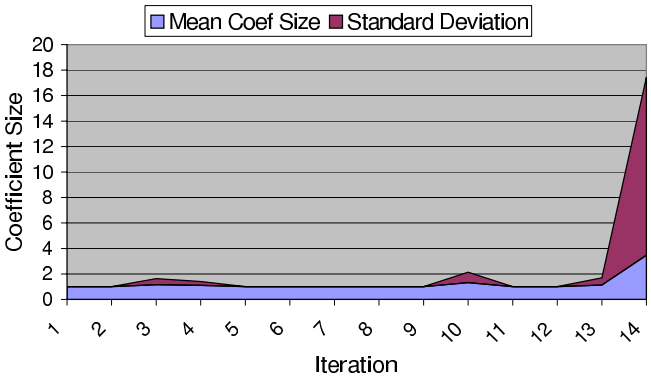

Figure 12: nw04: 87482 integer variables, 100% gap closed by split closure

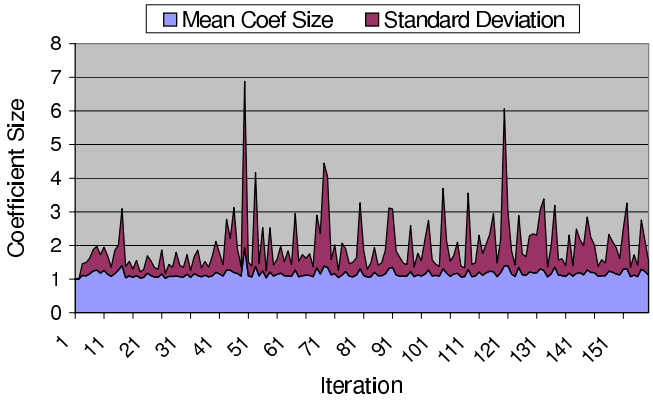

Figure 13: air05: 7195 integer variables, 62.05% gap closed by split closure

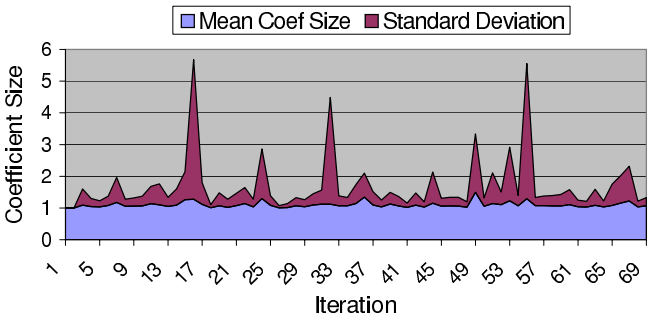

Figure 14: seymour: 1372 integer variables, 61.94% gap closed by split closure

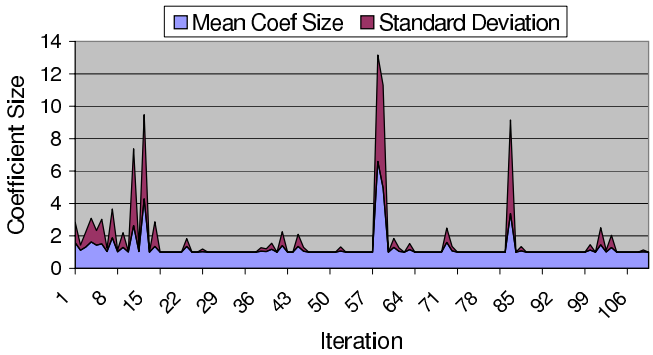

Figure 15: misc03: 159 integer variables, 51.47% gap closed by split closure

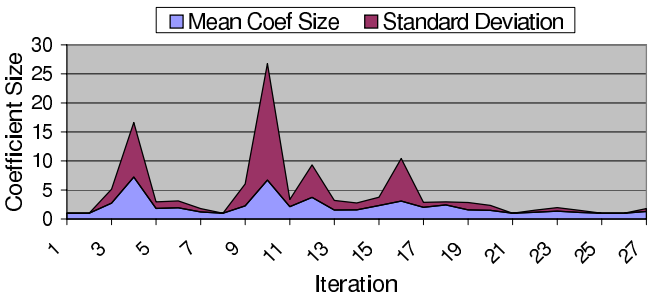

Figure 16: p0033: 33 integer variables, 87.42% gap closed by split closure

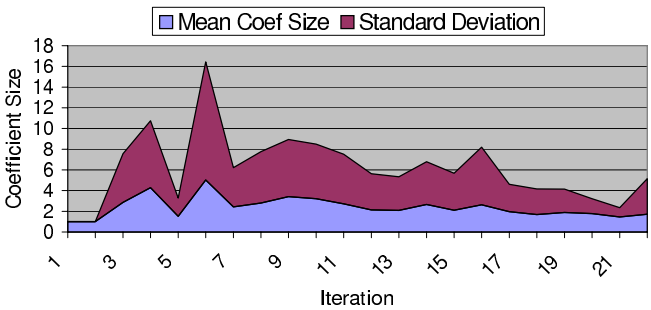

Figure 17: qnet1_o: 1417 integer variables, 100% gap closed by split closure

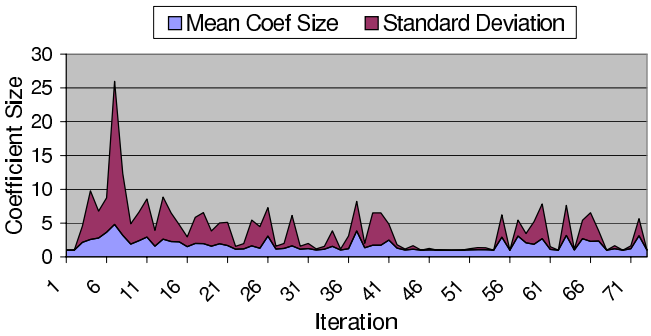

Figure 18: gesa2_o: 720 integer variables, 100.00% gap closed by split closure

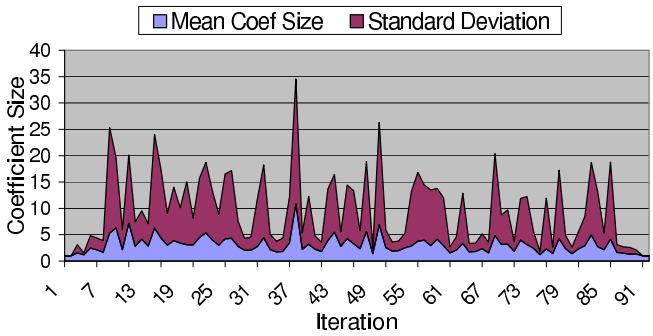

Figure 19: arki001: 538 integer variables, 83.05% gap closed by split closure

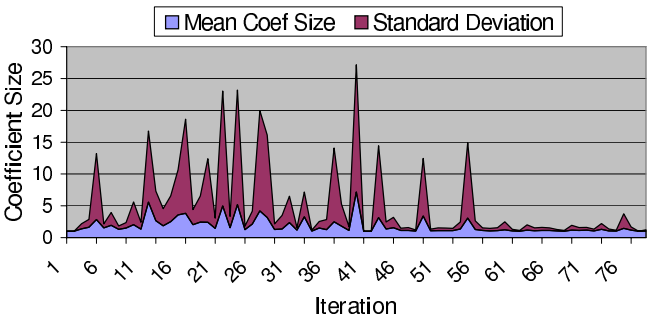

Figure 20: vpm1: 168 integer variables, 100% gap closed by split closure

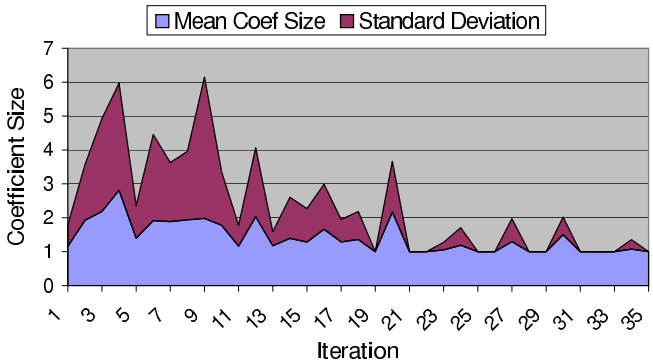

Figure 21: pp08aCUTS: 64 integer variables, 97.01% gap closed by split closure

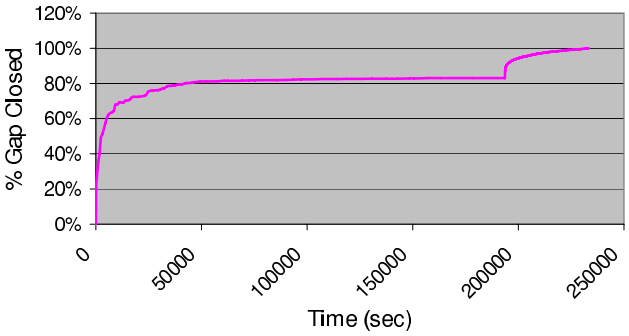

Figure 22: Percentage duality gap closed w.r.t time (strengthened formulation: preprocessing + rank-1 split cuts + CPLEX 9.0)

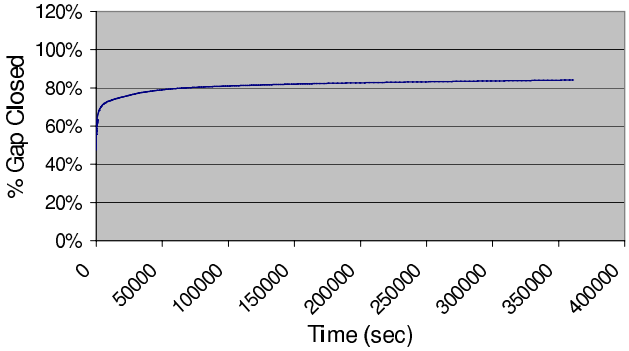

Figure 23: Percentage duality gap closed w.r.t time (preprocessing + CPLEX 9.0)

References

[1] K. Andersen, G. Cornuejols and Y. Li, "Split Closure and Intersection Cuts." *Mathematical Programming A, 102*, 2005, 457-493.

[2] E. Balas, "Disjunctive Programming." *Annals of Discrete Mathematics, 5,* 1979, 3-51.

[3] E. Balas, S. Ceria and G. Cornuéjols, "A Lift-and-Project cutting Plane Algorithm for Mixed 0-1 Programs." *Mathematical Programming, 58,* 1993, 295-324.

[4] E. Balas, S. Ceria and G. Cornuéjols, "Mixed 0-1 Programming by Lift-and-Project in a Branch-and-Cut Framework. *Management Science, 42,* 1996, 1229-1246.

[5] E. Balas and C. de Souza, "The Vertex Separator Problem: A Polyhedral Investigation." *Mathematical Programming, 103:3*, 2005, 583-608.

[6] E. Balas and M. Perregaard, "Lift and Project for Mixed 0-1 Programming: Recent Progress." *Discrete Applied Mathematics, 123,* 2002, 129-154.

[7] E. Balas and M. Perregaard, "A Precise Correspondence Between Lift-and-Project Cuts, Simple Disjunctive Cuts, and Mixed Integer Gomory Cuts for 0-1 Programming." *Mathematical Programming B, 94,* 2003, 221-245.

[8] E. Balas and A. Saxena, "Separation Functions in Disjunctive Programming." In preparation.

[9] J.E Beasley, OR-Library, `people.brunel.ac.uk/~mastjjb/jeb/info.html`.

[10] P. Bonami, G. Cornuéjols, S. Dash, M. Fischetti and A. Lodi, "Projected Chvátal-Gomory cuts for Mixed Integer Linear Programs." Paper presented at the 10th Combinatorial Optimization Workshop (Aussois, France), January 2006.

[11] A. Caprara and A. Letchford, "On the separation of split cuts and related inequalities." *Mathematical Programming 94,* 2003, 279-294.

[12] COIN, Computational infrastructure for operations research, `http://www.coin-or.org`

[13] W.J. Cook, R. Kannan and A. Schrijver, "Chvatal Closures for Mixed Integer Programming Problems." *Mathematical Programming, 47,* 1990, 155-174.

[14] G. Cornuéjols and Y. Li, "Elementary Closures for Integer Programs." *Operations Research Letters, 28*, 2001, 1-8.

[15] S. Dash, O. Günlük and A. Lodi, "Optimizing over the MIR closure." Talk presented at INFORMS Meeting (San Francisco), November 2006.

[16] C. de Souza and E. Balas,"The Vertex Separator Problem: Algorithms and Computations." *Mathematical Programming, 103:3*, 2005, 609-631.

[17] M. Fischetti and A. Lodi, "Optimizing over the first Chvátal closure." Integer Programming and Combinatorial Optimization (M. Juenger and V. Kaibel eds.), *Lecture Notes in Computer Science, 3509*, Springer-Verlag Berlin Heidelberg, 12-22, 2005.

[18] J.L. Nazareth, "The Homotopy Principle and Algorithms for Linear Programming," *SIAM Journal on Optimization, 1,* 1991, 316-332.