A Computational Study of the Cutting Plane Tree Algorithm for General Mixed-Integer Linear Programs

Binyuan Chen
Department of Systems and Industrial Engineering
University of Arizona
bychen@email.arizona.edu
Simge Küçükyavuz, Suvrajeet Sen*
Data Driven Decisions Laboratory
Department of Integrated Systems Engineering
The Ohio State University
{kucukyavuz.2,sen.22}@osu.edu

October 17, 2011

Abstract

The cutting plane tree algorithm provides a finite procedure to solve general mixed-integer linear programs with bounded integer variables. The computational evidence provided in this paper illustrates that this algorithm is powerful enough to close a significant fraction of the integrality gap for moderately sized MIPLIB instances.

Keywords: Cutting plane algorithm, multi-term disjunctions.

1 Introduction

In a recent paper, Chen et al. [12] proposed a finite disjunctive characterization for MILP-G with bounded integer variables. Based on this characterization, the authors develop a cutting plane tree (CPT) algorithm to solve mixed-integer linear programs with general integer variables (MILP-G). The CPT algorithm provides a pure cutting plane method that solves MILP-G in finitely many steps, without resorting to reformulating the problem. Prior to this, Balas [3] shows that for MILP-G sequential convexification with respect to its integer variables (n_1 in all) may not give the convex hull in n_1 iterations. Owen and Mehrotra [18] show that repeated sequential convexification of MILP-G converges to the convex hull after infinitely many iterations.

The overarching theme of the CPT algorithm is to discover a polyhedral approximation that has the same optimal value as the original MILP-G. Such approximations not only have theoretical value in establishing that MILP-G can be solved by a pure cutting plane algorithm in finitely many steps, but also have important consequences in the design and analysis of decomposition methods for multi-stage and stochastic MILP-G [21]. The main ingredient of the CPT algorithm is the manner in which multi-term disjunctions are 'discovered' as the algorithm proceeds. Perregaard and Balas [19] and Sherali et al. [22] studied multi-term disjunctions for mixed binary linear programs (MILP-B) by using a fixed number of variables to form disjunctive sets to convexify. Multi-term disjunctions

*Corresponding Author: Suvrajeet Sen, 1971 Neil Avenue, Columbus, OH 43210, sen.22@osu.edu

can also be created in the context of a branch-and-bound search as considered for MILP-B in Balas [3] and by Sherali and Smith [24]. Despite the recent explosion of computational experiments using disjunctive programming, such as Balas and Bonami [4], Balas and Saxena [7], Bonami and Minoux [10], Fischetti et al. [16], there is a clear paucity of computational experience with multi-term disjunctions to solve MILPs, let alone, adaptive disjunctions of the type generated by the CPT algorithm. For recent computational experience with other classes of cutting planes such as mixed-integer rounding and Gomory cuts, we refer the reader to Dash et al. [15], and Zanette et al. [25], Dash and Goycoolea [14] and references therein.

A recent paper by Jörg [17] addresses the finiteness of disjunctive programming for general MILP. However, the algorithm requires the identification of all alternative optima in one phase, as well as the identification of all extreme directions in another, and hence it may not be practicable. Similarly, Adams and Sherali [2] propose a finite convex hull characterization for MILP-G based on Lagrange interpolation polynomials. Although their result establishes the existence of a finite convex hull representation for MILP-G, to the best of our knowledge, computational studies using this technique have not yet been reported.

In this paper, we study whether the finitely convergent CPT algorithm can be realized computationally. In order to verify the effectiveness of the CPT algorithm, there are in fact several choices that one has control over. a) the sequence in which multi-term disjunctions are explored algorithmically, b) the exact formulation to be used for the multi-term cut generation LP (CGLP), c) the manner in which the multi-term CGLP is solved, and d) the normalization scheme used in the CGLP. Computational tests that cover all these aspects of the CPT require a much more comprehensive investigation and are beyond the scope of this short paper. Our goals for this paper are limited to demonstrating that unlike other finitely convergent cutting plane algorithms for MILP-G, the CPT algorithm can provide a computationally interesting alternative to the standard two-term disjunctions. Moreover, we do address issue d) by identifying a particular normalization that appears to be effective within this context. In essence, this paper is a computational companion of the earlier paper of Chen et al. [12], which was largely a theoretical contribution.

The paper is organized as follows. In §2, we describe the implementation of the cutting plane tree algorithm. In §3, we present computational results using a significant number of instances from various MIPLIB libraries [8, 9, 1]. In order to isolate and identify the potential of our scheme of generating multi-term disjunctions, we do not include other computational devices such as branch-and-bound, other classes of cutting planes, heuristics or preprocessing strategies. We conclude in §4 with future research directions.

2 Cutting Plane Tree Algorithm

In this section, we summarize a slight generalization of the CPT algorithm of [12] in which multiple cuts are generated using those variables that are fractional in any iteration. This approach, which results in a "round of cuts" has become standard for computational experiments with lift-and-project cuts [5, 10] as well as Gomory cuts with multiple source rows.

The pseudo-code of the cutting plane tree algorithm with rounds of cuts, including the notation, is given in Algorithm 1. Comparing the pseudo-code presented in Algorithm 1 with that presented in [12], the main differences are due to the implementation of the rounds of cuts in line 18 and lines 5–13. The "for" loop in line 18 lets us generate multiple cuts. Lines 5–13 perform a specific tree management rule, which creates two new nodes, l_σ and r_σ, in the CPT emanating from the current node σ, provided that one of the fractional variables x_j that was used to form a disjunction in the previous iteration $(k-1)$ is integral in iteration k. In effect, the expansion of the tree is delayed

until one round of cuts renders a fractional variable integral.

Algorithm 1 Cutting plane tree algorithm

1: Initialization: $k = 1$, create a node o, where $p_o = null$. Let $\mathcal{T} = \{o\}$ be the current CPT, $\mathcal{L}_k := \{o\}$, be the set of leaf nodes of CPT at iteration k, and $\mathcal{C}_o = \{x | x_j \in [\ell_j, u_j], j = 1, \ldots, n_1\}$. Also let $\mathcal{L}(t)$ be the set of leaf nodes on subtree rooted at node t, $X_k = X_L \cap \mathcal{C}_o$ and $x^k \in \arg\min_{x \in X_k} c^T x$, where x^k is a vertex of X_k. Let l_σ, r_σ and p_σ denote the left child, right child and parent nodes of node σ, respectively. For a non-leaf node σ of $\mathcal{T}$, let $v_\sigma \in \{1, 2, \ldots, n_1\}$ be the index of the integer variable that is split, and an integer q_σ be the (lower) level of the splitting at node σ (i.e., $x_{v_\sigma} \leq q_\sigma$ for l_σ and $x_{v_\sigma} \geq q_\sigma + 1$ for r_σ).
2: **while** $X_k \neq \emptyset$ and $x_j^k \notin \mathbb{Z}$ for some $j \in \{1, \ldots, n_1\}$ **do**
3: Search for node $\sigma \in \mathcal{T}$ such that: either $\sigma \in \mathcal{L}_k$ and $x^k \in \mathcal{C}_\sigma$; or, node $\sigma \notin \mathcal{L}_k$ and $x^k \in \mathcal{C}_\sigma$, $x^k \notin \mathcal{C}_{l_\sigma}$ and $x^k \notin \mathcal{C}_{r_\sigma}$.
4: **if** $\sigma \in \mathcal{L}_k$ **then**
5: **if** $\exists j = 1, \ldots, n_1$ with $x_j^{k-1} \notin \mathbb{Z}$ and $x_j^k \in \mathbb{Z}$ **then**
6: Let $v_\sigma = j$, $\mathcal{L}_{k+1} \leftarrow \mathcal{L}_k \setminus \{\sigma\}$.
7: **Form_disjunction**$(X_k, \mathcal{T}, \mathcal{L}_{k+1}, \sigma, j, x^{k-1})$.
8: **if** $x_j^k \leq q_\sigma$ **then**
9: $\sigma \leftarrow l_\sigma$.
10: **else**
11: $\sigma \leftarrow r_\sigma$.
12: **end if**
13: **end if**
14: **else if** $\sigma \notin \mathcal{L}_k$ and $x^k \in \mathcal{C}_\sigma$, $x^k \notin \mathcal{C}_{l_\sigma}$ and $x^k \notin \mathcal{C}_{r_\sigma}$ **then**
15: Let $\mathcal{L}_{k+1} \leftarrow \mathcal{L}_k$.
16: **end if**
17: $X_{k+1} \leftarrow X_k$.
18: **for all** $j = 1, \ldots, n_1$ such that $x_j^k \notin \mathbb{Z}$ **do**
19: Let $\sigma_j \leftarrow \sigma, v_{\sigma_j} = j$, $q_{\sigma_j} = \lfloor x_j^k \rfloor, \mathcal{T}^j \leftarrow \mathcal{T}, \mathcal{L}_k^j \leftarrow \mathcal{L}_{k+1}, \mathcal{L}_{k+1}^j \leftarrow \mathcal{L}_k^j \setminus \mathcal{L}(\sigma_j)$.
20: **Form_disjunction**$(X_k, \mathcal{T}^j, \mathcal{L}_{k+1}^j, \sigma_j, j, x^k)$.
21: Generate a valid inequality for the set $\text{clconv}\{\cup_{t \in \mathcal{L}_{k+1}^j} (X_{k+1}) \cap \mathcal{C}_t)\}$ that cuts off x^k (using CGLP). Append this valid inequality to the set X_{k+1}.
22: **end for**
23: Let $k \leftarrow k + 1$. Let $x^k \in \arg\min_{x \in X_k} c^\top x$, where x^k is a vertex of X_k.
24: **end while**

2.1 Cut Generation

Here we describe line 21 of Algorithm 1 in more detail. Let x^k be the fractional solution at iteration k. For the variable $x_j^k, j \in \{1, \ldots, n_1\}$, which is used to expand the CPT to yield the set of leaf nodes $\mathcal{L}_{k+1}^j$, define the sets $X_k \cap \mathcal{C}_t^j = \{x | A_k x \geq b_k, L_t \leq x \leq U_t\}$ for $t \in \mathcal{L}_{k+1}^j$, where A_k and b_k are the cut-enhanced matrix and right-hand-side vector of MILP-G in iteration k, respectively, and L_t and U_t are the lower and upper bound vectors for x in leaf node t, respectively. Let the multipliers used in the cut generation linear program (CGLP) [3, 23] be $\{\lambda_t, \mu_t, \nu_t\}_{t \in \mathcal{L}_{k+1}^j}$, where λ_t corresponds to the vector of multipliers associated with $A_k x \geq b_k$, and μ_t, ν_t denote the vectors of multipliers for the lower and upper bound constraints, $L_t \leq x \leq U_t$, respectively. The CGLP has

Algorithm 2 function: **Form_disjunction** $(X, \mathcal{T}, \mathcal{L}, \sigma, j, \bar{x})$

Create a node l^- with $l_\sigma = l^-$, $p_{l^-} = \sigma$ and a node l^+ with $r_\sigma = l^+$, $p_{l^+} = \sigma$.
Let $\mathcal{C}_{l^-} = \{x \in \mathcal{C}_\sigma | x_j \leq \lfloor \bar{x}_j \rfloor\}$ and $\mathcal{C}_{l^+} = \{x \in \mathcal{C}_\sigma | x_j \geq \lceil \bar{x}_j \rceil\}$.
if $\mathcal{C}_{l^-} \cap X \neq \emptyset$ **then**
  Let $\mathcal{L} \leftarrow \mathcal{L} \cup \{l^-\}$. Let $\mathcal{T} \leftarrow \mathcal{T} \cup \{l^-\}$.
else
  Let $l^- = null$ (fathom).
end if
if $\mathcal{C}_{l^+} \cap X \neq \emptyset$ **then**
  Let $\mathcal{L} \leftarrow \mathcal{L} \cup \{l^+\}$. Let $\mathcal{T} \leftarrow \mathcal{T} \cup \{l^+\}$.
else
  Let $l^+ = null$ (fathom).
end if

the form:

$$\max \quad \pi_0 - \pi^\top x^k \tag{1}$$

$$\text{s.t.} \quad \pi = \lambda_t^\top A_k + \mu_t - \nu_t \qquad t \in \mathcal{L}_{k+1}^j \tag{2}$$

$$\pi_0 \leq \lambda_t^\top b_k + \mu_t^\top L_t - \nu_t^\top U_t \quad t \in \mathcal{L}_{k+1}^j \tag{3}$$

$$(\lambda, \mu, \nu) \geq \mathbf{0}. \tag{4}$$

The optimal solution to the CGLP yields an inequality $\pi^\top x \geq \pi_0$ valid for X that cuts off x^k.

The feasible set defined by (2)–(4) is a cone, and its extreme rays correspond to valid inequalities including all facets of the disjunctive program at hand. To truncate the feasible set and obtain a bounded solution, various normalization constraints have been studied in the literature [3, 4, 6, 11, 13, 16, 20]. The introduction of the normalization constraint may generate extreme points of CGLP that do not correspond to facets of the convex hull (closure) of the disjunctive set. The computational results presented in Balas et al. [5], Fischetti et al. [16], Balas and Bonami [4], as well as the results summarized in Section 3, show that the choice of normalization has some impact on the performance and stability of the algorithm. Recognizing this, we experiment with two alternatives, although others are clearly possible [4, 16].

1. Weighted Cut Coefficients (WCC). This normalization constraint is a variant of that of Balas and Perregaard [6], with $w^\top \pi = 1$, where $w = \hat{x} - x^k$, and $\hat{x}$ is some feasible solution to $\text{clconv}\{\cup_{t \in \mathcal{L}_{k+1}^j} (X_k \cap \mathcal{C}_t)\}$. For example, [10] choose $\hat{x}$ to be a mixed-integer feasible solution to X. Since the latter would require our experiments to use either a pre-processing function, or a branch-and-cut algorithm, we instead choose $\hat{x}$ to be an optimal extreme point of either $X_k \cap \mathcal{C}_{l_\sigma}$ or $X_k \cap \mathcal{C}_{r_\sigma}$, whichever gives a better bound. As shown in [6], including the constraint $w^\top \pi = 1$ to (1)–(4) renders the CGLP objective bounded.

2. Minimum 1-Norm Cut (M1NC) Another way to ensure that the CGLP is bounded is to translate the coordinates to the point x^k, and then to find a minimum norm hyperplane that separates the origin from the closure of the convex hull of the disjunctive set. As a result, such cuts may be referred to as "deep" cuts [13, 11, 20]. As one can recognize, this idea can be traced back to the very origins of optimization, i.e., separating hyperplane theorems. However, care must be taken to implement such ideas in a computationally effective way. It turns out that the choice of the norm used in defining the projection is critical to the success of a computer

implementation of the CGLP. Computational experiments reported in [11] suggest that cuts which minimize the ℓ_2 norm do not lead to an effective computational approach. We have confirmed this conclusion in our own experiments. One of the normalizations that we report in Section 3 is based on choosing cut coefficients that minimize the ℓ_1 norm. The resulting CGLP in the translated space, at iteration k for fractional variable $x_j^k, j \in \{1, \ldots, n_1\}$, is

$$\begin{array}{lll} \min & \sum_{j=1}^{n} |\pi_j| & \\ \text{s.t.} & \pi = \lambda_t^\top A_k + \mu_t - \nu_t & t \in \mathcal{L}_{k+1}^j \\ & \lambda_t^\top B_k + \mu_t^\top L_t^k - \nu_t^\top U_t^k \geq 1 & t \in \mathcal{L}_{k+1}^j \\ & (\lambda, \mu, \nu) \geq \mathbf{0} & \end{array}$$

where $B_k = b_k - A_k x^k$, $L_t^k = L_t - x^k$ and $U_t^k = U_t - x^k$. The cut that we obtain is $\pi^\top (x - x^k) \geq 1$. It can be shown that the problem of obtaining a cut with minimum ℓ_1 norm is the dual of the problem of projecting the origin onto the convex hull (closure) of a disjunctive set using the ℓ_∞ norm.

3 Computational Results

The instances for this study were selected from the MIPLIB 2.0, 3.0 and 2003 libraries [8, 9, 1]. As in Bonami and Minoux [10], we limited our test instances to those with less than 1000 variables. Out of these, only `stein15` belongs exclusively to MIPLIB 2.0. The characteristics of our test instances can be found at `http://miplib.zib.de/`. The computer code was implemented in C in Microsoft Visual Studio 2003. We conducted our experiments on Windows XP platform with Intel Q9450 Core 2 Quad processor, with 4 cores, 4 threads, running at 2.66 GHz speed with 4 GB of RAM. The linear programming solver is IBM ILOG CPLEX 12.2.

In Tables 1 and 2, we report our computational experiments using CPT with multiple cuts per iteration (one for each fractional variable) for the previously identified set of MILP-G and MILP-B instances, respectively. We report performance of various algorithms based on one hour (CPU time) of computation:

1. The two normalization variants (WCC and M1NC) as described in Section 2.1,
2. Within each variant we compare the number of cuts as well as percentage gap closure due to cuts obtained by using multi-term disjunctions with those from using two-term (elementary) disjunctions (as in lift-and-project cuts).

Table 1 reports `Cuts` (the number of cuts added for each variant), T (the number of disjunctions or leaf nodes of CPT at termination), N (the total number of nodes in the CPT at termination), `%gapcl` (the percentage integrality gap closed by using either multi-term or two-term disjunctions). The comparison with two-term disjunctions based on cuts and gap closure is provided as a benchmark against which we evaluate the potential effectiveness of multi-term disjunctions within CPT. The integrality gap closed is calculated from the bound available after the last iteration, which is given by $(f^* - lp)/(\tilde{f} - lp) \times 100\%$, where lp is the objective function value of the linear relaxation of the instance, $\tilde{f}$ is the known integer optimal solution, and f^* is the bound from the specific variant at termination.

We use the following stopping rule: $\min\left\{x_j^k - \left\lfloor x_j^k \right\rfloor, \left\lceil x_j^k \right\rceil - x_j^k\right\} \leq \epsilon$ for all $j = 1, \ldots, n_1$. The value of ϵ is also used to select the set of variables for which disjunctive cuts are generated, i.e.,

if $\min\left\{x_j^k - \left\lfloor x_j^k \right\rfloor, \left\lceil x_j^k \right\rceil - x_j^k\right\} > \epsilon$, then we say that $x_j \notin \mathbb{Z}$. Hence, a higher value of ϵ lets the algorithms choose variables with greater fractionality for cut generation. For this reason we choose $\epsilon = 10^{-8}$ for both WCC and M1NC. For some instances, we had to tune the value of ϵ to prevent early termination due to numerical issues. These instances, indicated with a † in the tables, are summarized below.

- For WCC, we use $\epsilon = 10^{-4}$ for instances `bell5, gen, modglob, set1ch`, $\epsilon = 10^{-3}$ for `blend2` and $\epsilon = 10^{-2}$ for instances `fixnet6, pp08a`.
- For M1NC, we use $\epsilon = 10^{-4}$ for the instance `modglob`.

Note that M1NC has fewer instances of numerical instability than WCC.

In examining the outcomes reported in Table 1, we first compare specific CGLP normalizations for multi-term disjunctions. Observe that the average gap closures for WCC and M1NC are similar, although M1NC outperforms WCC ("wins") in 7 out of 10 instances. We also observe that for the same amount of computational time, M1NC generates more cuts for every instance. Indeed, in all cases reported in Table 1, M1NC generated over thrice as many cuts as the WCC formulation of CGLP. This suggests that the CGLP formulations using M1NC may be much easier to solve.

Table 1 also provides comparisons between multi-term and two-term disjunctions for each variant of the CGLP. Using average gap closure as a measure of performance, the CPT algorithm with multiple disjunctions with the WCC normalization has a slight edge over the corresponding results for two-term disjunctions in Table 1. This statistic can be traced to its significantly better performance (by 30% for WCC) for `blend2`. However, the win-loss record on the basis of gap closure is in favor of two-term disjunctions (6-3). Because of their relatively smaller CGLPs, more two-term cuts are generated within an hour than multi-term cuts. Despite this obvious advantage of two-term disjunctions, the overall average gap closure for multi-term disjunctions is quite similar. Since the latter is likely to provide more compact approximations, they may be useful for cases in which the size of the LP approximation is important (e.g., in stochastic mixed-integer programming).

Next, we examine the computations reported in Table 2. As in Table 1, this table reports the same metrics for MILP-B instances in MIPLIB. We should mention, however, that 100% gap closure was attained before the time limit for a few instances. For these instances, we report the solution time in parentheses (in seconds), instead of the 100% gap closure. As with the results reported in Table 1 the average gap closure of multi-term disjunctions for current implementation of WCC is slightly superior to the average gap closure provided by corresponding two-term disjunctions, although the win-loss record once again favors two-term disjunctions (15-9). Moreover, the trend (of generating many more cuts in M1NC) continues in Table 2, although there are some instances (e.g. `glass4, mas74, mas76, modglob, rgn, and vpm1`) in which WCC generates more cuts. Overall, the average gap closure obtained with M1NC is around 4% more than that obtained with WCC (in Table 2). Moreover, if one examines the difference in gap closures at the 10%, 20%, and 50% levels in Table 2, we observe the win-loss data favors M1NC as follows (with M1NC wins listed first): (9-3) (6-1),(1-1). Unlike the negative computational experience using the 2-norm for CGLP [11], our experience with CGLPs with the 1-norm (i.e. M1NC) appears to be encouraging. On the other hand, one of the wins for WCC is a difficult instance called `glass4`; WCC was able to make significant progress, getting 75% gap closure, before succumbing to some numerical difficulties, indicated with an asterisk *.

Table 1: Results for CPT on General Integer Instances from MIPLIB

	WCC						M1NC					
	Multi-term				Two-term		Multi-term				Two-term	
Instance	Cuts	T	N	%gapcl	Cuts	%gapcl	Cuts	T	N	%gapcl	Cuts	%gapcl
bell3a	74	5	9	70.7%	684	71.1%	1320	9	17	74.6%	1023	72.4%
bell5	503	2	4	94.3%†	1398	96.4%	1089	4	6	97.2%	1690	97.4%
blend2	440	3	5	65.0%†	1195	37.5%	2551	2	3	41.5%	2550	42.6%
flugpl	1342	3	5	23.6%	727	22.9%	8991	4	8	27.1%	9816	28.7%
gen	698	3	6	91.9%†	636	86.3%	1881	2	3	96.1%	1870	93.8%
gt2	663	3	5	97.1%	1278	98.8%	1958	14	27	93.2%	1709	100%
noswot	315	7	13	0.0%	329	0.0%	904	5	9	0.0%	1061	0.0%
rout	732	2	3	3.1%	807	6.5%	2030	2	3	7.0%	2053	7.8%
timtab1	1132	2	3	37.5%	1240	40.0%	3018	3	5	44.0%	3131	46.9%
timtab2	1170	2	3	29.4%	1384	31.5%	2818	2	3	33.9%	2876	34.7%
Average	706.9	3.2	5.6	51.3%	967.8	49.1%	2656.0	4.7	8.4	51.5%	2777.9	52.4%

Table 2: Results for CPT on Mixed Binary MIPLIB Instances

	WCC						M1NC					
	Multi-term				Two-term		Multi-term				Two-term	
Instance	Cuts	T	N	%gapcl	Cuts	%gapcl	Cuts	T	N	%gapcl	Cuts	%gapcl
aflow30a	794	2	3	42.9%	761	37.6%	1456	2	3	70.8%	1598	76.4%
danoint	398	2	3	2.6%	541	3.4%	500	5	9	1.7%	864	2.6%
dcmulti	975	2	3	99.5%	958	96.1%	1390	4	8	(2397s)	1168	(1283s)
egout	173	2	3	(7s)	107	(1s)	120	3	5	(2s)	83	(1s)
enigma	1524	4	7	nogap	2044	nogap	3324	9	18	nogap	4358	nogap
fixnet6	675	2	3	69.1%†	654	84.4%	1132	4	7	90.9%	1717	90.9%
glass4	192	2	3	75.0%*	186	14.9%*	12	4	7	25.0%	28	0.0%
lseu	2068	2	3	47.6%	2447	48.6%	2983	3	5	56.9%	2931	62.3%
markshare1	399	11	21	0.0%	3354	0.0%	2358	25	49	0.0%	5134	0.0%
markshare2	185	8	15	0.0%	3236	0.0%	2475	20	39	0.0%	4267	0.0%
mas74	14	2	3	11.0%	9	11.9%	0	1	1	0.0%	0	0.0%
mas76	4	3	5	7.0%	2	4.9%	0	1	1	0.0%	0	0.0%
misc03	1862	3	5	56.0%	1884	60.1%	8755	2	3	57.3%	9220	58.7%
misc07	1412	3	5	12.0%	1444	10.9%	4859	4	7	12.1%	5810	12.9%
mod008	1413	3	5	25.0%	1556	27.0%	1710	2	3	31.6%	1579	29.4%
modglob	619	2	3	85.7%†	508	87.3%	479	2	4	99.9%†	474	99.4%
opt1217	920	2	3	0.6%	756	19.0%	1087	11	21	0.5%	3093	0.8%
p0033	280	3	5	72.9%	797	73.2%	5203	6	11	99.4%	4660	100%
p0201	974	2	3	94.7%	1115	95.2%	2463	2	3	76.6%	2425	79.2%
p0282	1753	2	3	97.1%	1484	97.1%	2664	2	3	98.3%	2803	98.4%
p0548	1038	2	3	46.79%	1047	53.1%	1305	2	3	100.0%	1330	(302s)
pk1	969	7	13	0.0%	2190	0.0%	3373	6	11	0.0%	3436	0.0%
pp08a	1169	2	3	98.1%†	889	97.6%	2355	2	3	99.5%	2426	99.6%
pp08aCUTS	519	2	3	89.3%	1587	94.0%	2178	2	3	98.9%	2279	98.7%
qiu	144	1	1	56.0%	148	56.0%	876	2	3	81.4%	862	84.9%
rgn	1144	3	5	46.5%	1681	53.2%	435	5	9	53.6%	2485	67.6%
set1ch	1840	2	4	100%†	1405	99.6%	1675	5	14	(2896s)	1371	(308s)
stein15	3914	2	3	24.2%	4037	26.1%	4406	2	3	38.5%	4413	38.8%
stein27	4010	2	3	12.9%	4039	13.2%	4344	2	3	24.6%	4351	24.6%
stein45	3051	2	3	0.0%	3088	0.0%	3507	2	3	0.0%	3526	2.9%
vpm1	1485	2	3	84.9%	1339	81.1%	1304	2	3	(206s)	1368	(229s)
vpm2	1445	2	3	79.5%	991	78.9%	1723	2	3	88.1%	2026	89.1%
Average	1167.6	2.8	4.7	51.1%	1446.4	49.3%	2201.6	4.6	8.4	55.0%	2565.2	55.6%

We also tested this version of our algorithm on larger instances. However, due to the increased size of the CGLPs, the algorithm does not proceed deeper into the CPT in most instances. Therefore, not much additional strength can be obtained by considering multi-term disjunctions if a one hour time limit is enforced and the CGLPs are solved without taking advantage of their structure.

4 Concluding Remarks

This computational study reveals that the CPT algorithm leads to stronger cuts so that fewer cuts are generated within an hour to achieve a similar gap closure when compared to two-term disjunctions. Also, the choice of the normalization to use in the CGLP makes a difference. In particular, the performance (numerical stability and percentage gap closure) favors M1NC over WCC. There are several factors that decide the effectiveness of cutting plane algorithms based on multi-term disjunctions discovered via the CPT algorithm. These include rules governing the sequence of multi-term disjunctions used for cut formation, the CGLP formulation, the manner in which CGLPs are solved, as well as the normalization to use. Our computational study suggests that further investigations, covering other combinations of choices, would be worthwhile.

5 Acknowledgments

We are grateful to Dinakar Gade for his assistance in the computational experiments and for his comments on an earlier version of this paper. We also thank the AE and the referee for the suggestions that improved the paper. Binyuan Chen was supported, in part, by Air Force Office of Scientific Research (AFOSR) Grant F49620-03-1-0377; Simge Küçükyavuz is supported, in part, by NSF-CMMI Grants 0917952 and 1100383; and Suvrajeet Sen is supported, in part, by AFOSR Grant: FA9950-08-1-0154, and NSF-CMMI Grants 0900070 and 1100383.

References

[1] Achterberg, T., Koch, T., and Martin, A. (2006). MIPLIB 2003. *Operations Research Letters*, 34(4):361–372.

[2] Adams, W. P. and Sherali, H. D. (2005). A hierarchy of relaxations leading to the convex hull representation for general discrete optimization problems. *Annals of Operations Research*, 140(1):21–47.

[3] Balas, E. (1979). Disjunctive programming. *Annals of Discrete Mathematics*, 5:3–51.

[4] Balas, E. and Bonami, P. (2009). Generating lift-and-project cuts from the LP simplex tableau: open source implementation and testing of new variants. *Mathematical Programming Computation*, 1:165–199.

[5] Balas, E., Ceria, S., and Cornuéjols, G. (1996). Mixed 0-1 programming by lift-and-project in a branch-and-cut framework. *Management Science*, 42(9):1229–1246.

[6] Balas, E. and Perregaard, M. (2002). Lift-and-project for mixed 0-1 programming: recent progress. *Discrete Applied Mathematics*, 123:129–154.

[7] Balas, E. and Saxena, A. (2008). Optimizing over the split closure. *Mathematical Programming*, 113:219–240.

[8] Bixby, R. E., Boyd, E. A., and Indovina, R. R. (1992). MIPLIB: A test set of mixed integer programming problems. *SIAM News*, 25:16.

[9] Bixby, R. E., Ceria, S., McZeal, C. M., and Savelsbergh, M. W. P. (1996). An updated mixed integer linear programming library: MIPLIB 3.0. `http://www.caam.rice.edu/~bixby/miplib/miplib.html`.

[10] Bonami, P. and Minoux, M. (2005). Using rank-1 lift-and-project closures to generate cuts for 0-1 MIPs, a computational investigation. *Discrete Optimization*, 2:288–307.

[11] Cadoux, F. (2010). Computing deep facet-defining cuts for mixed-integer programming. *Mathematical Programming*, 122:197–223.

[12] Chen, B., Küçükyavuz, S., and Sen, S. (2011). Finite disjunctive programming characterizations for general mixed-integer linear programs. *Operations Research*, 59(1):202–210.

[13] Cornuéjols, G. and Lemaréchal, C. (2006). A convex-analysis perspective on disjunctive cuts. *Mathematical Programming*, 106:567–586.

[14] Dash, S. and Goycoolea, M. (2010). A heuristic to generate rank-1 GMI cuts. *Mathematical Programming Computation*, 2(3–4):231–257.

[15] Dash, S., Günlük, O., and Lodi, A. (2010). MIR closures of polyhedral sets. *Mathematical Programming*, 121(1):33–60.

[16] Fischetti, M., Lodi, A., and Tramontani, A. (2009). On the separation of disjunctive cuts. *Mathematical Programming*, 28(1–2):1–26.

[17] Jörg, M. (2007). k-disjunctive cuts and a finite cutting plane algorithm for general mixed integer linear programs. `http://arxiv.org/PS_cache/arxiv/pdf/0707/0707.3945v1.pdf`.

[18] Owen, J. H. and Mehrotra, S. (2001). A disjunctive cutting plane procedure for general mixed-integer linear programs. *Mathematical Programming*, 89:437–448.

[19] Perregaard, M. and Balas, E. (2001). Generating cuts from multiple-term disjunctions. In *Proceedings of the 8th International IPCO Conference on Integer Programming and Combinatorial Optimization*, pages 348–360, London, UK. Springer-Verlag.

[20] Rey, P. A. and Sagastizábal, C. A. (2002). Convex normalizations in lift-and-project methods for 0-1 programming. *Annals of Operations Research*, 116:91–112.

[21] Sen, S. and Sherali, H. D. (2006). Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming. *Mathematical Programming*, 106:203–223.

[22] Sherali, H. D., Lee, Y., and Kim, Y. (2005). Partial convexification cuts for 0-1 mixed-integer programs. *European Journal of Operational Research*, 165(3):625–648.

[23] Sherali, H. D. and Shetty, C. M. (1980). *Optimization with Disjunctive Constraints*. Springer Verlag, Berlin.

[24] Sherali, H. D. and Smith, J. C. (2010). Higher-level RLT or disjunctive cuts based on a partial enumeration strategy for 0-1 mixed-integer programs. Online first.

[25] Zanette, A., Fischetti, M., and Balas, E. (2011). Lexicography and degeneracy: can a pure cutting plane algorithm work? *Mathematical Programming*, 130(1):153–176.