An in-out approach to disjunctive optimization

Matteo Fischetti[1] and Domenico Salvagnin[2]

[1] DEI, University of Padova, Italy
email: matteo.fischetti@unipd.it
[2] DMPA, University of Padova, Italy
email: salvagni@math.unipd.it

Abstract. Cutting plane methods are widely used for solving convex optimization problems and are of fundamental importance, e.g., to provide tight bounds for Mixed-Integer Programs (MIPs). This is obtained by embedding a cut-separation module within a search scheme. The importance of a sound search scheme is well known in the Constraint Programming (CP) community. Unfortunately, the "standard" search scheme typically used for MIP problems, known as the Kelley method, is often quite unsatisfactory because of saturation issues.
In this paper we address the so-called *Lift-and-Project closure* for 0-1 MIPs associated with all disjunctive cuts generated from a given set of elementary disjunction. We focus on the search scheme embedding the generated cuts. In particular, we analyze a general meta-scheme for cutting plane algorithms, called in-out search, that was recently proposed by Ben-Ameur and Neto [1]. Computational results on test instances from the literature are presented, showing that using a more clever meta-scheme on top of a black-box cut generator may lead to a significant improvement.

Keywords: Mixed-integer programming, cutting planes, disjunctive optimization.

1 Introduction

Cutting plane methods are widely used for solving convex optimization problems and are of fundamental importance, e.g., to provide tight bounds for Mixed-Integer Programs (MIPs). These methods are made by two equally important components: (i) the separation procedure (oracle) that *produces* the cut(s) used to tighten the current relaxation, and (ii) the overall search framework that actually *uses* the generated cuts and determines the next point to cut.

In the last 50 years, a considerable research effort has been devoted to the study of effective families of MIP cutting planes, as well as to the

definition of sound separation procedures and cut selection criteria [2,3]. However, the search component was much less studied, at least in the MIP context where one typically cuts a vertex of the current LP relaxation, and then reoptimizes the new LP to get a new vertex to cut—a notable exception is the recent paper [4] dealing with Benders' decomposition. The resulting approach—known as "the Kelley method" [5]—can however be rather inefficient, the main so if the separation procedure is not able to produce strong (e.g., facet defining or, at least, supporting) cuts. As a matter of fact, alternative search schemes are available that work with non-extreme (internal) points [6,7], including the famous ellipsoid [8,9] and analytic center [10–12] methods; we refer the reader to [13] for an introduction. The convergence behavior of these search methods is less dependant on the quality of the generated cuts, which is a big advantage when working with general MIPs where separation procedures tend to saturate and to produce shallow cuts. A drawback is that, at each iteration, one needs to recompute a certain "core" point, a task that can be significantly more time consuming than a simple LP reoptimization. An interesting hybrid search method, called *in-out search*, was recently proposed by Ben-Ameur and Neto [1].

In this paper we address disjunctive optimization [14] in the MIP context. It essentially consists of a cutting plane method where cuts are separated by exploiting a given set of valid disjunctions. In particular, we consider 0-1 MIPs and the associated *Lift-and-Project closure*, defined by all the disjunctive cuts that can be derived from the "elementary" set of disjunctions of the type $x_j \leq 0$ or $x_j \geq 1$ for each integer-constrained variable x_j. This topic is currently the subject of intensive investigation by the Mathematical Programming community. Our current research topic is in fact to move the research focus from the widely investigated separation module to the search scheme where the generated cuts are actually embedded. A first step in this direction is reported in the present paper, where we investigate the use of disjunctive cuts within an in-out search shell. Computational results show that the resulting scheme outperforms the standard one, in that it produces tighter bounds within shorter computing times and need much fewer cuts—though they use exactly the same separation module.

2 In-out search

Let us consider a generic MIP of the form

$$\min\{c^T x : Ax = b,\ l \leq x \leq u,\ x_j \in \mathbb{Z}\ \forall j \in I\}$$

and let $P := \{x \in \mathbb{R}^n : Ax = b,\ l \leq x \leq u\}$ denote the associated LP relaxation polyhedron. In addition, let us assume the oracle structure allows one to define a "cut closure", P_1, obtained by intersecting P with the half-spaces induced by all possible inequalities returned by the oracle. Cutting plane methods are meant to compute $z_1 := \min\{c^T x : x \in P_1\}$, with P_1 described implicitly through the oracle.

In-out search works with two points: an "internal" (possibly non optimal) point $q \in P_1$, and an optimal vertex x^* of P (possibly not in P_1). By construction, the final (unknown) value z_1 belongs to the *uncertainty interval* $[c^T x^*, c^T q]$, i.e., at each iteration both a lower and an upper bound on z_1 are available. If the two points q and x^* coincide, the cutting plane method ends. Otherwise, we apply a bisection step over the line segment $[x^*, q]$, i.e., we invoke the separation procedure in the attempt of cutting the middle point $y := (x^* + q)/2$. (In the original proposal, the separation point is more generally defined as $y := \alpha x^* + (1 - \alpha)q$ for a given $\alpha \in (0, 1]$.) If a violated cut is returned, we add it to the current LP that is reoptimized to update x^*, hopefully reducing the current lower bound $c^T x^*$. Otherwise, we update $q := y$, thus improving the upper bound and actually *halving* the current uncertainty interval.

The basic scheme above can perform poorly in its final iterations. Indeed, it may happen that x^* already belongs to P_1, but the search is not stopped because the internal point q is still far from x^*. We then propose a simple but quite effective modification of the original scheme where we just count the number of consecutive updates to q, say k, and separate directly x^* in case $k > 3$. If the separation is unsuccessful, then we can terminate the search, otherwise we reset counter k and continue with the usual strategy of cutting the middle point y.

As to the initialization of $q \in P_1$, this is a simple task in many practical settings, including the MIP applications where finding a feasible integer solution q is not difficult in practice.

3 Disjunctive cuts

Consider the generic MIP of the previous section. To simplify notation, we concentrate on 0-1 MIPs where $l_j = 0$ and $u_j = 1$ for all $j \in I$. Our order of business is to optimize over the Lift-and-Project closure, say P_1, obtained from P by adding all linear inequalities valid for $P^j := conv(\{x \in P : x_j \leq 0\} \cup \{x \in P : x_j \geq 1\})$ for $j \in I$. To this end, given a point $x^* \in P$ (not necessarily a vertex), for each $j \in I$ with $0 < x^*_j < 1$ we construct a certain *Cut Generation Linear Program* (CGLP)

whose solution allows us to detect a valid inequality for P^j violated by x^* (if any). Various CGLPs have been proposed in the literature; the one chosen for our tests has a size comparable with that of the original LP, whereas other versions require to roughly double this size. Given x^* and a disjunction $x_j \leq 0 \vee x_j \geq 1$ violated by x^*, our CGLP reads:

$$\max x_j - d^* \tag{1}$$
$$Ax = d^* b \tag{2}$$
$$d^* l \leq x \leq d^* l + (x^* - l) \tag{3}$$
$$d^* u - (u - x^*) \leq x \leq d^* u \tag{4}$$

where $d^* = x_j^* > 0$ (the two sets of bound constraints can of course be merged). Given the optimal dual multipliers $(\lambda, -\sigma'', \sigma', -\tau', \tau'')$ associated with the constraints of the CGLP, it is possible to derive a most-violated disjunctive cut $\gamma x \geq \gamma_0$, where $\gamma = \sigma' - \tau' - u_0 e_j$, $\gamma_0 = \sigma' l - \tau' u$, and $u_0 = 1 - \lambda b - (\sigma' - \sigma'') + (\tau' - \tau'')u$.

4 Computational results (sketch)

We implemented both the standard (`kelley`) and in-out (`in-out`) separation schemes and we compared them on a collection of 50 0-1 MIP instances from MIPLIB 3.0 [15] and 2003 [16], and on 15 set covering instances from ORLIB [17]. We used IBM ILOG Cplex 11.2 as black-box LP solver, and to compute a first heuristic solution to initialize the in-out internal point q. Both schemes are given a time limit of 1 hour, and generate only one cut at each iteration–taken from the disjunction associated to the most fractional variable. Cumulative results are reported in Table 1, where `time` denotes the geometric mean of the computing times (CPU seconds on an Intel Q6600 PC running at 2.4 GHz), `itr` denotes the geometric mean of the number of iterations (i.e., cuts), `cl.gap` denotes the average gap closed w.r.t the best known integer solution, and `L&P cl.gap` denotes the average gap closed w.r.t. the best known upper bound on z_1 (this upper bound is obtained as the minimum between the best-known integer solution value and the last upper bound on z_1 computed by the in-out algorithm). The results clearly show the effectiveness of in-out search, in particular for set covering instances.

References

1. Ben-Ameur, W., Neto, J.: Acceleration of cutting-plane and column generation algorithms: Applications to network design. Networks **49**(1) (2007) 3–17

testbed	method	time (s)	itr	cl.gap	L&P cl.gap
MIPLIB	kelley	38.18	1,501	40.8%	63.7%
	in-out	28.42	592	41.2%	64.1%
set covering	kelley	2,281.60	16,993	35.2%	71.8%
	in-out	757.29	1,575	38.7%	85.8%

Table 1. Cumulative results on Lift-and-Project optimization

2. Cornuéjols, G.: Valid inequalities for mixed integer linear programs. Mathematical Programming **112**(1) (2008) 3–44
3. Cornuéjols, G., Lemaréchal, C.: A convex analysis perspective on disjunctive cuts. Mathematical Programming **106**(3) (2006) 567–586
4. Naoum-Sawaya, J., Elhedhli, S.: An interior-point branch-and-cut algorithm for mixed integer programs. Technical report, Department of Management Sciences, University of Waterloo (2009)
5. Kelley, J.E.: The cutting plane method for solving convex programs. Journal of the SIAM **8** (1960) 703–712
6. Elzinga, J., Moore, T.J.: A central cutting plane algorithm for the convex programming problem. Mathematical Programming **8** (1975) 134–145
7. Ye, Y.: Interior Point Algorithms: Theory and Analysis. John Wiley, New York (1997)
8. Tarasov, S., Khachiyan, L., Erlikh, I.: The method of inscribed ellipsoids. Soviet Mathematics Doklady **37** (1988) 226–230
9. Bland, R.G., Goldfarb, D., Todd, M.J.: The ellipsoid method: a survey. Operations Research **29**(6) (1981) 1039–1091
10. Atkinson, D.S., Vaidya, P.M.: A cutting plane algorithm for convex programming that uses analytic centers. Mathematical Programming **69** (1995) 1–43
11. Nesterov, Y.: Cutting plane algorithms from analytic centers: efficiency estimates. Mathematical Programming **69**(1) (1995) 149–176
12. Goffin, J.L., Vial, J.P.: On the computation of weighted analytic centers and dual ellipsoids with the projective algorithm. Mathematical Programming **60** (1993) 81–92
13. Boyd, S., Vandenberghe, L.: Localization and cutting-plane methods. available at `http://www.stanford.edu/class/ee364b/notes/localization_methods_notes.pdf` (2007)
14. Balas, E.: Disjunctive programming. Annals of Discrete Mathematics **5** (1979) 3–51
15. Bixby, R.E., Ceria, S., McZeal, C.M., Savelsbergh, M.W.P.: An updated mixed integer programming library: MIPLIB 3.0. Optima **58** (1998) 12–15 See `http://www.caam.rice.edu/bixby/miplib/miplib.html`.
16. Achterberg, T., Koch, T., Martin, A.: MIPLIB 2003. Operations Research Letters **34**(4) (2006) 1–12 See `http://miplib.zib.de`.
17. Beasley, J.: OR-Library: distributing test problems by electronic mail. Journal of the Operational Research Society **41**(11) (1990) 1069–1072 See `http://people.brunel.ac.uk/~mastjjb/jeb/info.html`.