Mixed-Integer Programming: State-of-the-art and Fashionable Topics

Andrea Lodi*

We consider a general *Mixed Integer Linear Program* (MIP) in the form

$$\min\{c^T x : Ax \geqslant b, x \geqslant 0,\ x_j \in \mathbb{Z}\ \forall j \in I\} \tag{1}$$

where we do not assume that the matrix A has any special structure. Thus, the algorithmic approach relies on the iterative solution, through general-purpose techniques, of the *Linear Programming* (LP) relaxation

$$\min\{c^T x : Ax \geqslant b, x \geqslant 0\}, \tag{2}$$

i.e., the same as problem (1) above but the integrality requirement on the x variables in the set I has been dropped. We denote an optimal solution of problem (2) as x^*. The reason for dropping such constraints is that MIP is $\mathbb{NP}$-hard while LP is polynomially solvable and general-purpose techniques for its solution are efficient in practice.

In these lectures we do not cover LP state-of-the-art, while we cover the basic characteristics and components of current, commercial and non-commercial, MIP solvers. However, Bixby et al. [2] report that in 2004 an LP was solved, by CPLEX 8, a million times faster than it was by CPLEX 1 in 1990, three orders of magnitudes due to hardware and to software improvements, respectively. This gives a clear indication of how much LP technology has been and is important for MIP development.

Roughly speaking, using the LP computation as a tool, MIP solvers integrate the *branch-and-bound* and the *cutting plane* algorithms through variations of the general *branch-and-cut* scheme proposed by Padberg & Rinaldi [15, 16] in the context of the *Traveling Salesman Problem* (TSP).

The branch-and-bound algorithm, Land & Doig [11]. In its basic version the branch-and-bound algorithm iteratively partitions the solution space into sub-MIPs (the children nodes) which have the same theoretical complexity of the originating MIP (the father node, or the root node if it is the initial MIP). Usually, for MIP solvers the branching creates two children by using the rounding of the solution of the LP relaxation value of a fractional variable, say x_j, constrained to be integral

$$x_j \leqslant \lfloor x_j^* \rfloor \quad \textbf{OR} \quad x_j \geqslant \lfloor x_j^* \rfloor + 1. \tag{3}$$

The two children above are often referred to as *left* (or "down") branch and *right* (or "up") branch, respectively. On each of the sub-MIPs the integrality requirement on the variables $x_j, \forall j \in I$ is relaxed and the LP relaxation is solved. Despite the theoretical complexity, the sub-MIPs become smaller and smaller due to the partition mechanism (basically some of the decisions are taken) and eventually the LP relaxation is directly integral for all the variables in I. In addition, the LP relaxation is solved at every node to decide if the node itself is worthwhile to be further partitioned: if the LP relaxation value is already not smaller than the best feasible solution encountered so far, called *incumbent*, the node can safely be fathomed because none of its children will yield a better solution than the incumbent. Finally, a node is also fathomed if its LP relaxation is infeasible.

*DEIS, University of Bologna, Viale Risorgimento 2, 40136 Bologna, Italy, andrea.lodi@unibo.it

The cutting plane algorithm, Gomory [9]. Any MIP can be solved without branching by simply finding its "right" linear programming description, more precisely, the *convex hull* of its (mixed-)integer solutions. In order to do that, one has to iteratively solve the so called *separation problem*

> Given a feasible solution x^* of the LP relaxation (2) which is not feasible for the MIP (1), find a linear inequality $\alpha^T x \geqslant \alpha_0$ which is valid for (1), i.e., satisfied by all feasible solutions $\bar{x}$ of the system (1), while it is violated by x^*, i.e., $\alpha^T x^* < \alpha_0$.

Any inequality solving the separation problem is called a *cutting plane* (or a *cut*, for short) and has the effect of tightening the LP relaxation to better approximate the convex hull.

Gomory [9] has given an algorithm that converges in a finite number of iterations for pure *Integer Linear Programming* (IP)[1] with integer data. Such an algorithm solves the separation problem above in an efficient and elegant manner in the special case in which x^* is an optimal basis of the LP relaxation. No algorithm of this kind is known for MIPs, that being one of the most intriguing open questions in the area (see, e.g., Cook, Kannan & Schrijver [3]).

The idea behind integrating the two algorithms above is that LP relaxations (2) do not naturally well approximate, in general, the convex hull of (mixed-)integer solutions of MIPs (1). Thus, some extra work to devise a better approximation by tightening any relaxation with additional linear inequalities (cutting planes) increases the chances that fewer nodes in the search tree are needed. On the other hand, pure cutting plane algorithms show, in general, a slow convergence and the addition of too many cuts can lead to very large LPs which in turn present numerical difficulties for the solvers. The branch-and-cut algorithm has been proven to be very effective initially for combinatorial optimization problems (like TSP) with special-purpose cuts based on a polyhedral analysis and later on in the general MIP context.

Lecture 1. In the first lecture, we discuss the evolution of MIP solvers having in mind both a performance perspective and a modeling/application viewpoint. We initially present some important MIP milestones with no aim of being exhaustive with respect to algorithms and software. We then go into the details of the basic components of MIP codes. Then, we describe some important tools that allow a relevant degree of flexibility in the development of MIP-based applications. Finally, we discuss the challenges for the next generation MIP solvers by first presenting a list of difficult MIP classes on which better performance/strategies would be extremely beneficial. This lecture is largely based on the paper [12].

Lecture 2. In the second lecture, we discuss both branching and cutting at an advanced level. Specifically, we review the attempts to branch on disjunctions more complicated than (3) (see, [10, 14, 13]) and we consider the recent and very intriguing idea of generating cuts by using more than one row of the simplex tableau (see, [1, 8, 4, 5, 6, 7].

References

[1] K. Andersen, Q. Louveaux, R. Weismantel, and L.A. Wolsey. Inequalities from two rows of a simplex tableau. In M. Fischetti and D.P. Williamson, editors, *Integer Programming and Combinatorial Optimization - IPCO 2007*, volume 4513 of *Lecture Notes in Computer Science*, pages 1–15, Berlin Heidelberg, 2007. Springer-Verlag.

[2] R.E. Bixby, M. Fenelon, Z. Gu, E. Rothberg, and R. Wunderling. Mixed-integer programming: A progress report. In M. Grötschel, editor, *The Sparpest Cut: The Impact of Manfred Padberg and his Work*, pages 309–325. MPS-SIAM Series on Optimization, 2004.

[1] IPs are the special case of MIPs where all variables belong to I, i.e., are constrained to be integer.

[3] W.J. Cook, R. Kannan, and A. Schrijver. Chvátal closures for mixed integer programming problems. *Mathematical Programming*, 47:155–174, 1990.

[4] G. Cornuéjols and F. Margot. On the facets of mixed integer programs with two integer variables and two constraints. *Mathematical Programming*, 120:429–456, 2009.

[5] S. Dey and L.A. Wolsey. Lifting integer variables in minimal inequalities corresponding to lattice-free triangles. In A. Lodi, A. Panconesi, and G. Rinaldi, editors, *Integer Programming and Combinatorial Optimization - IPCO 2008*, volume 5035 of *Lecture Notes in Computer Science*, pages 463–475. Springer-Verlag, Berlin Heidelberg, 2008.

[6] S. S. Dey and A. Tramontani. Recent developments in multi-row cuts. *Optima*, 80:2–8, 2009.

[7] S.S. Dey, A. Lodi, A. Tramontani, and L.A. Wolsey. Experiments with two row tableau cuts. Technical Report OR/09/11, DEIS, Università di Bologna, 2009.

[8] D.G. Espinoza. Computing with multi-row gomory cuts. In A. Lodi, A. Panconesi, and G. Rinaldi, editors, *Integer Programming and Combinatorial Optimization - IPCO 2008*, volume 5035 of *Lecture Notes in Computer Science*, pages 214–224. Springer-Verlag, Berlin Heidelberg, 2008.

[9] R.E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64:275–278, 1958.

[10] M. Karamanov and G. Cornuéjols. Branching on general disjunctions. Technical report, Tepper School of Business, CMU, 2005, revised 2008.

[11] A.H. Land and A.G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28:497–520, 1960.

[12] A. Lodi. Mip computation. In M. Jünger, T.M. Liebling, D. Naddef, G.L. Nemhauser, W.R. Pulleyblank, G. Reinelt, G. Rinaldi, and L.A. Wolsey, editors, *50 Years of Integer Programming 1958-2008*, pages 619–645. Springer-Verlag, 2009.

[13] A. Lodi, T.K. Ralphs, F. Rossi, and S. Smriglio. Interdiction branching. Technical Report OR/09/10, DEIS, Università di Bologna, 2009.

[14] A. Mahajan and T.K. Ralphs. Experiments with branching using general disjunctions. In *The Proceedings of the Eleventh INFORMS Computing Society Meeting*, pages 101–118, 2009.

[15] M.W. Padberg and G. Rinaldi. Optimization of a 532-city symmetric traveling salesman problem by branch and cut. *Operations Research Letters*, 6:1–7, 1987.

[16] M.W. Padberg and G. Rinaldi. A branch and cut algorithm for the resolution of large-scale symmetric traveling salesmen problems. *SIAM Review*, 33:60–100, 1991.