Branch and Cut[*]

John E. Mitchell[†]

May 12, 2010

Combinatorial optimization problems can often be formulated as mixed integer linear programming problems, as discussed in Section 1.4.1.1 in this encyclopedia. They can then be solved using *branch-and-cut*, which is an exact algorithm combining branch-and-bound (see Section 1.4.1.2 of this encyclopedia) and cutting planes (see Section 1.4.3 of this encyclopedia). The basic idea is to take a linear programming relaxation of the problem, solve the relaxation, and then either improve the relaxation by adding additional valid constraints, or split the problem into two or more subproblems and repeat the process.

Gomory first proposed strengthening linear programming relaxations of integer programming problems by incorporating extra constraints (or *cutting planes*) in the 1950's [28]. These cutting planes are derived from the optimal simplex tableau, so they are broadly applicable. However, they fell into disfavor for many years because they seemed to get stuck and run out of power. The cuts are now used in more sophisticated ways and are incorporated into the major commercial packages for integer programming. These packages also include several other families of general cutting planes. General cutting planes are discussed in section 1 of this entry and also in the entries in section 1.4.3 of the Encyclopedia.

Land and Doig [42] proposed a branch-and-bound approach in 1960. In branch-and-bound, the linear programming relaxation of the integer program is solved. If the solution is

[*]To appear in the Wiley Encyclopedia of Operations Research and Management Science

[†]Department of Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, New York 12180-1590, U.S.A. Email:mitchj@rpi.edu. The work of this author was supported in part by the National Science Foundation under grant DMS-0715446.

fractional then the problem is split into two subproblems and the process repeated, creating a tree of subproblems. The value of the linear programming relaxation gives a lower bound on the optimal value of the integer program at the corresponding node of the tree, for a minimization problem. If this lower bound is greater than the value of a known feasible solution then the node can be pruned, which greatly reduces the total size of the tree. Branch-and-bound became more popular than cutting plane methods for many years, because of the computational difficulties with the latter.

Interest in cutting planes resurfaced in the 1980's. Crowder et al. [18] showed the strength of general cuts obtained by regarding a single row of an integer program as a knapsack problem. In addition, the polyhedral theory of many classes of problems was derived, and this led to problem-specific cutting planes that were very successful, leading to great speed-ups in computational time when compared to using branch-and-bound. For many classes of problems (for example, the traveling salesman problem), an initial integer programming formulation contains a large number of constraints, possibly even an exponential number. In such a situation, it is not computationally attractive to explicitly include all of these constraints in the LP relaxation, and they can be added selectively as cutting planes. Problem specific cutting planes are the subject of section 2 of this entry, and polyhedral theory is the topic of section 1.4.3.1 of the Encyclopedia.

In theory, pure cutting plane methods can be used to solve integer programs, without the need to employ branching. In practice, cutting plane methods appear to tail-off, and so it becomes faster to combine together the two approaches. Initially, cutting planes were employed only at the root node the tree, in an approach now called cut-and-branch. Examples include [18] as well as work on the traveling salesman problem [19]. The set of cuts generated at the root node is not exhaustive, so it is possible that the subsequent branch-and-bound approach leads to an integer solution that is not actually feasible in the integer program. In such a situation, it is then necessary to add additional cuts and restart the process.

Later in the 1980's, cutting planes were employed throughout the tree. The best known of these results is for the traveling salesman problem, starting with the work of Padberg and

Rinaldi [61]; an excellent discussion of this problem is contained in the book by Applegate et al. [2]. Other notable early work includes the research of Grötschel, Jünger, and Reinelt on the linear ordering problem [32] and on the maximum cut problem [33]. In the 1990's, it was discovered that general cutting planes can actually be very effective in branch-and-cut approaches to integer programming problems, thanks to the work of Balas et al. [5, 6]. The integration of cutting planes with branch-and-bound is discussed in more detail in section 3 of this entry.

Refinements and extensions are discussed in section 4. For example, it is possible to generalize the branch-and-cut approach to solve mixed integer nonlinear programming problems and even nonlinear programs without integrality constraints. Also in this section, we consider exploitation of parallel computational hardware. In addition to parallel computers, it is common for problems to be solved on clusters of computers (including cloud computers) or on the multicore processors now frequent in desktop and even laptop computers. Branch-and-cut algorithms can be parallelized by solving different nodes of the tree on different processors.

The Lanchester Prize winning book by Nemhauser and Wolsey[58] contains an excellent discussion of polyhedral theory, integer programming, and branch-and-cut. Other very good and relevant books are those by Wolsey [69] and Lee [43]. The three-volume text by Schrijver [65] is also an excellent reference. Various surveys of branch-and-cut have appeared over the years, including [14, 39, 48, 53]. The aforementioned text on the traveling salesman problem by Applegate et al. [2] provides a very accessible development of integer programming, including branch-and-cut.

1 General cutting planes

Our standard form integer programming problem is the following:

$$\begin{array}{lrcl} \min & c^T x & & \\ \text{subject to} & Ax & \geq & b \\ & x & \geq & 0 \\ & x_i & & \text{integer } \forall i \in I \end{array} \qquad (ILP)$$

where x and c are n-vectors, b is an m-vector, A is an $m \times n$ matrix, and I is a subset of the indices $\{1, \ldots, n\}$. Any upper bound constraints on the variables are included in the inequality constraints $Ax \geq b$. The optimal value of (ILP) is denoted by z^*. Any feasible solution $\bar{x}$ to (ILP) provides an upper bound $c^T\bar{x}$ on the optimal value z^* of the problem. A lower bound can be obtained by solving a relaxation of (ILP). In this entry, we are concerned with linear programming (LP) relaxations, which are obtained by relaxing the integrality restriction.

The lower bound provided by the LP relaxation can be improved by tightening up the relaxation through the addition of valid linear constraints. Typically, these constraints are satisfied by all feasible solutions to (ILP) but violated by the optimal solution to the LP relaxation. The LP relaxation can be solved again after the addition of the constraints, and the process repeated.

Cutting planes are discussed in detail in entry 1.4.3 of this encyclopedia, and its subentries. In this section, we summarize some of the cutting planes that have been used to solve general integer programming problems, and which are now included in commercial integer programming packages (see, for example, Bixby and Rothberg [12] and Ashford [3]).

Gomory cuts are derived from a row of the optimal simplex tableau for the LP relaxation [29]. These were generalized by Chvátal [17], giving Chvátal-Gomory cuts, which can be derived from any nonnegative linear combination of the linear constraints of (ILP). For simplicity, we consider the case where all the variables are required to be integer. In particular, if $u \in I\!R^m$ is nonnegative then the constraint

$$u^T A x \geq u^T b$$

is valid for the LP relaxation of (ILP). Since $x \geq 0$, this constraint can be weakened to

$$\lceil u^T A \rceil x \geq u^T b$$

where $\lceil u^T A \rceil$ is the n-dimensional row vector obtained by rounding up each entry of the row vector $u^T A$. Since each entry of x is nonnegative, the left-hand side of this inequality must be integral for any feasible solution to (ILP). Hence, we can round up the right hand side to obtain the valid constraint

$$\lceil u^T A \rceil x \geq \lceil u^T b \rceil,$$

a Chvátal-Gomory cutting plane. Chvátal showed that any valid inequality for (ILP) can be obtained by repeatedly applying this rounding procedure [17].

Gomory cutting planes fell out of favor for many years, but computational results in the 1990's [6, 15] showed that they could be very helpful. According to [12], they are the most useful of the general cutting planes. Fischetti and Lodi [25] showed that just one round of generating every possible inequality from the original constraints $Ax \geq b$ can give a very good approximation to the convex hull of the set of feasible solutions. Letchford [44] described a method for generating deep Chvátal-Gomory cutting planes. One of the problems with Gomory cutting planes is that eventually dual degeneracy is encountered, which can lead to a basis matrix with a large condition number, if care is not taken in the generation of the cuts. Zanette et al. [71] demonstrated that employing lexicographic cut generation rules can lead to a set of Gomory cutting planes that interact well with one another and allow a pure cutting plane method to work effectively; see also the related paper [7]. Gomory cutting planes can also be derived for mixed integer programs; see Marchand and Wolsey [49] for some computational results. Chvátal-Gomory cutting planes are considered in far more detail in entry 1.4.3.5 of this Encyclopedia.

Cover inequalities are inequalities that are valid for knapsack problems. Crowder et al. [18] considered each row of an integer program as a separate knapsack problem, and then generated cover inequalities for the individual rows. They showed that this powerful technique could be employed in a cut-and-branch algorithm to solve general integer programming problems. Their approach has been considerably refined in recent years, becoming a standard part of branch-and-cut implementations. These inequalities are discussed in detail in entry 1.4.3.2 of this Encyclopedia.

The theory of disjunctive inequalities [4] gives a methodology for generating general cutting planes that can be powerful on certain hard integer programs [5, 16]. The theory was originally developed for binary variables and has been extended. Given a binary variable x_i, the process is to find the convex hull of two sets: the set of feasible points in the LP relaxation with $x_i = 0$, and the set of feasible points in the LP relaxation with $x_i = 1$. The process is then iterated over all the binary variables. The theory gives a method to systematically construct the convex hull of the set of feasible solutions to an integer program. Generation of a cut may require solution of a linear program, so in practice this method is used selectively. There are methods for generating disjunctive cuts using the optimal simplex tableau for the relaxation; see for example Balas and Perregaard [8, 63]. For more information see entries 1.4.3.8 and 1.4.4.1 of this encyclopedia.

Recently there has been interest in deriving cuts using two rows of the simplex tableau, which potentially could be stronger than cuts derived from just a single row. These methods use ideas from lattice theory and group theory. For more details see Andersen et al. [1] and also the survey article by Dey and Tramontani [22].

Computational experience with different classes of general cutting planes is detailed by Bixby and Rothberg [12]. One point the authors make is that different families of cutting planes can interact with each other, so the benefit of using all of several different families of cuts is not necessarily equal to the product of the benefits of using each family individually.

2 Problem specific cutting planes

Many combinatorial optimization problems can only be expressed as integer linear programming problems with an exponential number of constraints. For example, a standard formulation of the traveling salesman problem uses degree constraints to ensure each city is visited exactly once. It also requires the inclusion of subtour elimination constraints to ensure that any integral solution corresponds to a tour that connects all the cities, and the number of subtour elimination constraints is exponential in the number of cities. Thus, it is impractical to include all of these constraints in the integer programming formulation, and they should be added as cutting planes. The first demonstration of the strength of this cutting plane approach was by Dantzig et al. [20], who showed that a problem with 42 cities could be solved to optimality by adding just a limited number of subtour elimination constraints, and some other cutting planes. Cutting plane methods for the traveling salesman problem were revisited in the 1980's [31, 62], and subsequently the work of Applegate et al. [2] has realized an algorithm that can find provably optimal solutions to problems with as many as 85,900 cities. Their code Concorde is freely available.

The convex hull of the set of feasible integer solutions to a combinatorial optimization problem is a polyhedron. (See entry 1.4.3.1 on Basic Polyhedral Theory for more details.) If a linear programming description of this polyhedron is known then the problem can be solved effectively. However, the number of facets of the polyhedron is large (often exponential) for interesting combinatorial optimization problems. Thus, it is necessary to add the constraints selectively. The strongest cutting planes correspond to facets, and families of facets have been determined for many different problems. For example, the subtour elimination constraints mentioned earlier define facets.

Other problems for which cutting plane methods have been developed include the linear ordering problems [32, 34] with triangle inequalities and other classes of facet defining inequalities, the maxcut problem [33, 9, 10, 11, 21, 45, 52] with cycle-odd subset inequalities, matching problems [23, 30], clique and coloring problems [38, 51], fixed charge network flow problems [59], vehicle routing problems [36, 47, 56], and facility location problems [41].

Knowledge of a strong family of cutting planes is only useful in practice if effective separation routines are also developed, which can find violated constraints in the family efficiently. These separation routines can be simple or involved, even for the same class of constraints. For example, cycle-odd subset inequalities for maxcut problems can be checked by enumeration for all short cycles, but in order to guarantee that any violated inequality can be found it is necessary to use a max-flow algorithm for a graph derived from the original one [11]. Violated subtour elimination constraints for the traveling salesman problem can be found by searching for connected components in the solution to the LP relaxation, but this may not find all violated constraints, so more expensive routines have also been developed [2].

Let X be a feasibility integer program and let x be a point. The *separation problem* for X and x is to find a cutting plane that separates x from the convex hull of X, or determine that x is in this convex hull. If the separation problem for any point x can be solved in time no greater than $g(X)$, then problem X itself can be solved in time polynomial in $g(X)$ using the ellipsoid algorithm. This observation can be generalized as the equivalence of separation and optimization problems [35]. It follows that for an NP-Complete problem, it will not be possible to find a cutting plane for each point not in the convex hull in polynomial time (unless P=NP).

3 The Branch-and-Cut Algorithm

A branch-and-cut algorithm is outlined in Figure 3. The set of active nodes in the branch-and-cut tree is denoted by L. The value of the best known feasible point for (ILP) is stored as $\bar{z}$, and provides an upper bound on the optimal value of the integer program. This point is called the incumbent solution. We use $\underline{z}_l$ to denote a lower bound on the optimal value of the current subproblem l under consideration. This lower bound is initialized to the value of the parent node, and is then updated to the value of the LP relaxation of the subproblem.

Without the inclusion of Step 6, this becomes a branch-and-bound algorithm. A crucial point with branch-and-bound is that a subproblem l can be discarded once $\underline{z}_l \geq \bar{z}$, since it is then known that no feasible solution to the subproblem can be better than the incumbent solution. The other method for fathoming in Step 7 is when the optimal solution to the LP relaxation of the subproblem is feasible in the integer program, since this LP solution then solves the subproblem. We refer the reader to entry 1.4.1.2 of this encyclopedia for far more discussion of branch-and-bound, including preprocessing, options for branching, and reduced cost fixing and its exploitation.

The particular procedure employed in Step 5 can be a generic rounding procedure, or it

1. *Initialization*: Denote the initial integer programming problem by ILP^0 and define the set of active nodes to be $L = \{ILP^0\}$. Let $\bar{z} = +\infty$. Set $\underline{z}_l = -\infty$ for the initial problem $l \in L$.

2. *Termination*: If $L = \emptyset$, then STOP. If $\bar{z} = \infty$ then (ILP) is infeasible; else, the solution x^* which yielded the incumbent objective value $\bar{z}$ in Step 7(b) or Step 5 is optimal.

3. *Problem selection*: Select and delete a problem ILP^l from L.

4. *Relaxation:* Solve the LP relaxation of ILP^l. If the relaxation is infeasible, set $\underline{z}_l = +\infty$ and go to Step 7. If the relaxation is unbounded set $\underline{z}_l = -\infty$. If the relaxation has a finite optimal value let x^{lR} be an optimal solution and set $\underline{z}_l = c^T x^{lR}$.

5. *Heuristic Rounding:* If x^{lR} is not integral, and if desired, use a rounding approach or a heuristic approach to construct a feasible integral solution x^{lH}. Update $\bar{z} = \min\{c^T x^{lH}, \bar{z}\}$.

6. *Add cutting planes*: If desired, search for cutting planes that are violated by x^{lR}; if any are found, add them to the relaxation and return to Step 4.

7. *Fathoming and Pruning*:

   (a) *Fathom by bounds or infeasibility:* If $\underline{z}_l \geq \bar{z}$ go to Step 2.

   (b) *Fathom by integrality:* If $\underline{z}_l < \bar{z}$ and x^{lR} is integral feasible, update $\bar{z} = \underline{z}_l$, delete from L all problems with $\underline{z}_l \geq \bar{z}$, and go to Step 2.

8. *Partitioning*: Let $\{S^{lj}\}_{j=1,\ldots,k}$ be a partition of the constraint set S^l of problem ILP^l. Add problems $\{\text{ILP}^{lj}\}_{j=1,\ldots,k}$ to L, where ILP^{lj} is ILP^l with feasible region restricted to S^{lj}, and set $\underline{z}_{lj} = \underline{z}_l$ for $j = 1, \ldots, k$. Return to Step 2.

Figure 1: A general branch-and-cut algorithm

can be a rounding procedure modified to exploit the particular structure of the problem, or it can be a heuristic initiated either at the point x^{lR} or at a rounded version of this point. For more on heuristics see entry 1.4.1.8 of this encyclopedia.

The Step 6 decision of when to add cutting planes and when to branch can probably only be resolved through computational experimentation. The conclusion is dependent on the particular class of integer program, and on the types of cutting planes considered. Different types of cutting planes interact with each other, so care is needed in experimentation in order to determine reproducible benefits.

Cutting planes generated at one node of the tree may not be valid at another node. One option is to treat the cuts as local, and only use them for descendants of the node where they are generated. The disadvantage of this approach is that it becomes necessary to store several different sets of constraints, for different parts of the tree. Alternatively, the constraints can be modified to make them valid throughout the tree, using a process called *lifting*. In lifting, the value of the slack in the constraint is checked in the remainder of the tree, either by solving integer programs to get strong liftings or LP relaxations to get somewhat weaker lifted inequalities. Lifting is a general process for strengthening and modifying constraints and is discussed in entry 1.4.3.3 of this enclyclopedia.

One important aspect of a general integer programming code is preprocessing. One aspect of preprocessing is to tighten bounds on the variables and constraints by logical arguments and possibly solving LPs. This may lead to variable fixing or constraint elimination, and can have a dramatic impact on runtime (an average improvement of a factor of ten for the problems considered in [12]). Far more about preprocessing can be found in section 1.4.1.2 of this encyclopedia.

4 Refinements and extensions

Computational implementations of branch-and-cut are now very sophisticated and include many ideas from the research literature of the last 30 years. Commercial codes include CPLEX and GuRoBi [12], and XPRESS-MP [3]. Recent high-quality free software includes the COIN-OR branch-and-cut package Cbc [70], and the packages ABaCuS [40] and MINTO [57]. In this section, we consider some possible enhancements to current integer programming solvers.

Before discussing enhancements, we note that the computing environment is becoming ever more parallel. There have been sophisticated parallel computers for many years, and these have become more widespread, with local machines with at least 100 processors available to many users. In addition, there are now clusters of homogeneneous or heterogeneous processors linked together using software, there is the availability of cloud computing, and multicore processors are common in desktop and even laptop computers. This is an environment that must be exploited for a branch-and-cut implementation to remain competitive. Fortunately, the branching aspect of these algorithms leads to a natural way to parallelize: different subproblems in the tree are solved in different processors of the machine. Load balancing requires careful consideration, but in principle branch-and-cut algorithms should flourish in a world of parallel computers. For more concrete discussion of these issues, see [70].

A standard model for the traveling salesman problem is to use one variable for each edge, so if the graph has n vertices then there are $O(n^2)$ variables. Based just on the objective function coefficients, it is clear that the great majority of the variables can be (at least temporarily) fixed at zero. Thus, we can work with integer and linear programs where the number of variables is $O(n)$. Before fathoming any node of the tree, the eliminated edges can be checked using reduced costs, to see if they would have been helpful. This pricing step may lead to the introduction of variables, and the resulting algorithm is a form of branch-and-price-and-cut. For more on algorithms of this type, see entry 1.4.1.6 of this encyclopedia.

Many integer programming formulations possess a natural symmetry. For example, when scheduling jobs on several identical machines, the important decision is determining which set of jobs go together on a particular machine and then sequencing those jobs. Which machine performs which particular set of jobs doesn't matter. This poses difficulties for a standard branch-and-cut approach, because many variables have to be fixed in the branching tree before the symmetry is broken. There has been research on methods for breaking symmetry, using ideas from group theory and algebra [50, 60]. See entry 1.4.5.1 in this encyclopedia for more information.

Classically, cutting planes are satisfied by all feasible solutions. Cuts could potentially be strengthened by requiring only that all *optimal* solutions satisfy them. A similar possibility is noted in dual stabilization of column generation algorithms, see [46] for example.

Fischetti et al [27] note that it is possible to use sophisticated mixed integer programming techniques within a branch-and-cut solver. For example, the separation problems for some classes of cutting planes are themselves hard integer programs and so it is beneficial to use whatever MIP techniques are available in order to find strong cutting planes; see [38] for example. Construction of an initial feasible solution can also be performed by using integer programming techniques such as local branching [26].

It is superior to use interior point methods instead of the simplex method for some problems, at least in certain parts of the branch-and-cut process. Interior point methods have two potential advantages: first, cuts are generated from a more central solution, which leads to deeper cuts; secondly, interior point methods can solve large problems more quickly than simplex, and when many cuts are added at once the warm-start benefit enjoyed by simplex is no longer so advantageous. See [54] for a survey, and [52, 55] for computational results. Branch-and-cut can also be integrated with convex relaxations of the integer program, such as semidefinite relaxations; see [24, 37, 54, 64] for example.

Branch-and-cut algorithms have also been developed for mixed integer nonlinear programming problems. The cuts used in these approaches are typically disjunctive cuts. See for example Bonami et al. [13], which describes several different possible branch-and-cut approaches.

Branch-and-cut can even be used for problems without integrality restrictions. For example, Tawarmalani and Sahinidis [66, 67] describe an approach for global optimization of general nonlinear programs, and Vandenbussche and Nemhauser [68] show how branch-and-cut can be used to solve a nonconvex quadratic program by exploiting the optimality conditions.

5 Conclusions

The performance of branch-and-bound methods for integer programming has been dramatically improved by the inclusion of cutting planes, leading to branch-and-cut. The resulting exact methods have been successfully implemented in powerful general purpose solvers for mixed integer programs, and they are the method of choice for solving hard integer programs to optimality. The software has improved by several orders of magnitude in the last few years. Branch-and-cut solvers have also been developed for specific problems such as the traveling salesman problem, with such a code currently able to solve larger problems to optimality than any other approach. Branch-and-cut methods are still the subject of active research, with various ideas showing promise. The methods have also been extended to solve mixed integer nonlinear programs, and other classes of optimization problems.

References

[1] K. Andersen, Q. Louveaux, L. Wolsey, and R. Weismantel. Inequalities from two rows of a simplex tableau. *Lecture Notes in Computer Science*, 4513:1–15, 2007. Proceedings of IPCO 2007.

[2] D. Applegate, R. Bixby, V. Chvátal, and W. Cook. *The traveling salesman problem: a computational study.* Princeton University Press, Princeton, NJ, 2006.

[3] R. Ashford. Mixed integer programming: A historical perspective with xpress-mp. *Annals of Operations Research*, 149:5–17, 2007.

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

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

[6] E. Balas, S. Ceria, G. Cornuéjols, and N. Natraj. Gomory cuts revisited. *Operations Research Letters*, 19:1–9, 1996.

[7] E. Balas, M. Fischetti, and A. Zanette. On the enumerative nature of Gomory's dual cutting plane method. Technical report, DEI, Dipartimento di Ingegneria dell'Informazione, University of Padova, Italy, 2009.

[8] 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*, 94(2–3):221–245, 2003.

[9] F. Barahona, M. Grötschel, M. Jünger, and G. Reinelt. An application of combinatorial optimization to statistical physics and circuit layout design. *Operations Research*, 36(3):493–513, 1988.

[10] F. Barahona, M. Jünger, and G. Reinelt. Experiments in quadratic 0-1 programming. *Mathematical Programming*, 44(2):127–137, 1989.

[11] F. Barahona and A. R. Mahjoub. On the cut polytope. *Mathematical Programming*, 36:157–173, 1986.

[12] R. E. Bixby and E. Rothberg. Progress in computational mixed integer programming — a look back from the other side of the tipping point. *Annals of Operations Research*, 149:37–41, 2007.

[13] P. Bonami, L. T. Biegler, A. R. Conn, G. Cornuéjols, I. E. Grossmann, C. D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, and A. Wächter. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization*, 5(2):186–204, 2008.

[14] A. Caprara and M. Fischetti. Branch and cut algorithms. In M. Dell'Amico, F. Maffioli, and S. Martello, editors, *Annotated Bibliographies in Combinatorial Optimization*, chapter 4. John Wiley, 1997.

[15] S. Ceria, G. Cornuéjols, and M. Dawande. Combining and strengthening Gomory cuts. In E. Balas and J. Clausen, editors, *Lecture Notes in Computer Science*, volume 920, Heidelberg, 1995. Springer-Verlag.

[16] S. Ceria and G. Pataki. Solving integer and disjunctive programs by lift-and-project. In *Proceedings of the Sixth IPCO Conference*, 1998.

[17] V. Chvátal. Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete Mathematics*, 4:305–337, 1973.

[18] H. P. Crowder, E. L. Johnson, and M. Padberg. Solving large-scale zero-one linear programming problems. *Operations Research*, 31:803–834, 1983.

[19] H. P. Crowder and M. Padberg. Solving large-scale symmetric travelling salesman problems to optimality. *Management Science*, 26:495–509, 1980.

[20] G. B. Dantzig, D. R. Fulkerson, and S. M. Johnson. Solutions of a large-scale travelling salesman problem. *Operations Research*, 2:393–410, 1954.

[21] C. De Simone, M. Diehl, M. Jünger, P. Mutzel, G. Reinelt, and G. Rinaldi. Exact ground states of two-dimensional $\pm J$ Ising spin glasses. *Journal of Statistical Physics*, 84:1363–1371, 1996.

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

[23] J. Edmonds. Maximum matching and a polyhedron with 0, 1 vertices. *Journal of Research National Bureau of Standards*, 69B:125–130, 1965.

[24] I. Fischer, G. Gruber, F. Rendl, and R. Sotirov. Computational experience with a bundle approach for semidefinite cutting plane relaxations of max-cut and equipartition. *Mathematical Programming*, 105(2–3):451–469, 2006.

[25] M. Fischetti and A. Lodi. Optimizing over the first Chvátal closure. *Mathematical Programming*, 110(1):3–20, 2007.

[26] M. Fischetti and A. Lodi. Repairing MIP infeasibility through local branching. *Computers and Operations Research*, 35:1436–1445, 2008.

[27] M. Fischetti, A. Lodi, and D. Salvagnin. Just MIP it! *Annals of Information Systems*, 10:39–70, 2009.

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

[29] R. E. Gomory. An algorithm for integer solutions to linear programs. In R. L. Graves and P. Wolfe, editors, *Recent Advances in Mathematical Programming*, pages 269–302. McGraw-Hill, New York, 1963.

[30] M. Grötschel and O. Holland. Solving matching problems with linear programming. *Mathematical Programming*, 33:243–259, 1985.

[31] M. Grötschel and O. Holland. Solution of large-scale travelling salesman problems. *Mathematical Programming*, 51(2):141–202, 1991.

[32] M. Grötschel, M. Jünger, and G. Reinelt. A cutting plane algorithm for the linear ordering problem. *Operations Research*, 32:1195–1220, 1984.

[33] M. Grötschel, M. Jünger, and G. Reinelt. Calculating exact ground states of spin glasses: A polyhedral approach. In J. L. van Hemmen and I. Morgenstern, editors, *Proceedings of the Heidelberg Colloquium on Glassy Dynamics*, pages 325–353, 1985.

[34] M. Grötschel, M. Jünger, and G. Reinelt. Facets of the linear ordering polytope. *Mathematical Programming*, 33:43–60, 1985.

[35] M. Grötschel, L. Lovasz, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag, Berlin, Germany, 1988.

[36] A. Hadjar, O. Marcotte, and F. Soumis. A branch-and-cut algorithm for the multiple depot vehicle scheduling problem. *Operations Research*, 54(1):130–149, 2006.

[37] C. Helmberg and F. Rendl. Solving quadratic (0,1)-problems by semidefinite programs and cutting planes. *Mathematical Programming*, 82:291–315, 1998.

[38] X. Ji and J. E. Mitchell. Branch-and-price-and-cut on the clique partition problem with minimum clique size requirement. *Discrete Optimization*, 4(1):87–102, 2007.

[39] M. Jünger, G. Reinelt, and S. Thienel. Practical problem solving with cutting plane algorithms in combinatorial optimization. In *Combinatorial Optimization: DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, volume 20, pages 111–152. AMS, 1995.

[40] M. Jünger and S. Thienel. Introduction to ABACUS — A Branch-And-CUt System. *Operations Research Letters*, 22:83–95, 1998.

[41] M. Labbé, H. Yaman, and E. Gourdin. A branch and cut algorithm for hub location problems with single assignment. *Mathematical Programming*, 102(2):371–405, 2005.

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

[43] J. Lee. *A First Course in Combinatorial Optimization*. Cambridge University Press, Cambridge, 2004.

[44] A. N. Letchford. Totally tight Chvátal-Gomory cuts. *Operations Research Letters*, 30(2):71–73, 2002.

[45] F. Liers, M. Jünger, G. Reinelt, and G. Rinaldi. Computing exact ground states of hard Ising spin glass problems by branch-and-cut. In A. Hartmann and H. Rieger, editors, *New Optimization Algorithms in Physics*, pages 47–68. John Wiley, 2004.

[46] M. E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005.

[47] J. Lysgaard, A. N. Letchford, and R. W. Eglese. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, 100(2):423–445, 2004.

[48] H. Marchand, A. Martin, R. Weismantel, and L. A. Wolsey. Cutting planes in integer and mixed integer programming. *Discrete Applied Mathematics*, 123:397–446, 2002.

[49] H. Marchand and L. A. Wolsey. Aggregation and mixed integer rounding to solve MIPs. *Operations Research*, 49(3):363–371, 2001.

[50] F. Margot. Exploiting orbits in symmetric ILP. *Mathematical Programming*, 98(1–3):3–21, 2003.

[51] I. Méndez-Diaz and P. Zabala. A branch-and-cut algorithm for graph coloring. *Discrete Applied Mathematics*, 154(5):826–847, 2006.

[52] J. E. Mitchell. Computational experience with an interior point cutting plane algorithm. *SIAM Journal on Optimization*, 10(4):1212–1227, 2000.

[53] J. E. Mitchell. Branch-and-cut algorithms for combinatorial optimization problems. In P. M. Pardalos and M. G. C. Resende, editors, *Handbook of Applied Optimization*, pages 65–77. Oxford University Press, January 2002.

[54] J. E. Mitchell. Cutting plane methods and subgradient methods. In M. Oskoorouchi, editor, *TutORials in Operations Research*, chapter 2, pages 34–61. INFORMS, 2009.

[55] J. E. Mitchell and B. Borchers. Solving linear ordering problems with a combined interior point/simplex cutting plane algorithm. In H. L. Frenk *et al.*, editor, *High Performance Optimization*, chapter 14, pages 349–366. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2000.

[56] D. Naddef and G. Rinaldi. Branch-and-cut algorithms for the capacitated VRP. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, number 9 in Monographs on Discrete Mathematics and Applications, chapter 3, pages 53–84. SIAM, 2002.

[57] G. L. Nemhauser, M. W. P. Savelsbergh, and G. C. Sigismondi. MINTO, a Mixed INTeger Optimizer. *Operations Research Letters*, 15:47–58, 1994.

[58] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley, New York, 1988.

[59] F. Ortega and L. A. Wolsey. A branch-and-cut algorithm for the single-commodity, uncapacitated, fixed-charge network flow problem. *Networks*, 41(3):143–158, 2003.

[60] J. Ostrowski, J. Linderoth, F. Rossi, and S. Smriglio. Orbital branching. *Mathematical Programming*, accepted, online first.

[61] M. Padberg and G. Rinaldi. Optimization of a 532-city traveling salesman problem by branch and cut. *Operations Research Letters*, 6:1–8, 1987.

[62] M. Padberg and G. Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, 33(1):60–100, 1991.

[63] M. Perregaard. *Generating disjunctive cuts for mixed integer programs*. PhD thesis, Carnegie Mellon University, Graduate School of Industrial Administration, Pittsburgh, PA, September 2003.

[64] F. Rendl, G. Rinaldi, and A. Wiegele. Solving max-cut to optimality by intersecting semidefinite and polyhedral relaxations. *Mathematical Programming*, 121(2):307–335, 2010.

[65] A. Schrijver. *Combinatorial Optimization – Polyhedra and Efficiency*. Springer-Verlag, 2003.

[66] M. Tawarmalani and N. Sahinidis. *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*. Kluwer, Dordrecht, The Netherlands, 2002.

[67] M. Tawarmalani and N. Sahinidis. A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*, 103(2):225–249, 2005.

[68] D. Vandenbussche and G. L. Nemhauser. A branch-and-cut algorithm for nonconvex quadratic programs with box constraints. *Mathematical Programming*, 102(3):559–575, 2005.

[69] L. A. Wolsey. *Integer Programming.* John Wiley, New York, 1998.

[70] Y. Xu, T. K. Ralphs, L. Ladányi, and M. J. Saltzmann. Computational experience with a software framework for parallel integer programming. *INFORMS Journal on Computing*, 21(3):383–397, 2009.

[71] A. Zanette, M. Fischetti, and E. Balas. Lexicography and degeneracy: can a pure cutting plane algorithm work? *Mathematical Programming*, online first, 2010.