Computations in Mixed Integer Programming

A brief sum up of key concepts

Gabriele Dragotto

This document belongs is part proposal submitted in partial fulfillment of the requirements for the *Comprehensive Examination* for the Doctorate in *Mathematics Polytechnique Montreal*

Compiled March 13, 2021
Written in Februrary 2020

Contents

1 Mixed Integer Programming Computations

A general view From a high-level perspective[1], *MIP* computations are based on an expert blend of Land and Doig (1960) *branch-and-bound* algorithm, and the *cutting-plane* approach introduced by Gomory (1958). The *branch-and-bound* algorithm solves a series of *linear relaxation* of the original problem, elegantly splitting the domain of the latter through a series of *branching* decisions, and consequently solving restricted subproblems. The non-convexity due to the integer constraints is then handled indirectly by the *branching* scheme. Then, a *fathoming* procedure prunes nodes that cannot possibly improve an incumbent feasible solution, if any.

On the other hand, the *cutting-plane* approach iteratively solves a *linear relaxation* by adding valid cuts whenever a fractional solution violates an integrality constraint. The seminal work of Padberg and Rinaldi (1991) synthesized bounding and cutting into the so-called *branch-and-cut* algorithm (Algorithm 1), in the context of the Traveling Salesman Problem. This contribution showcased – for the first time – a computationally efficient combination of cutting planes, and the *branch-and-bound* algorithm. Since then, a considerable research effort has been made to bridge theory and computation in *MIP*, with a special focus on: (i) the underlying connections with polyhedral combinatorics, and the computational tractability of the different families of cutting planes, (ii) the effectiveness of a nourished toolkit of primal heuristics, (iii) the development of refined branching strategies, (iv) and, more recently, the integration with *ML*.

In what follows we consider *MIP*s in the form of

$$\max\{c^T x : x \in \mathcal{G}\} \tag{1}$$

$$\mathcal{G} := \{Ax \geq b,\ x \geq 0,\ x_i \in \mathbb{Z}\ \ \forall i \in \mathcal{I}\} \tag{2}$$

Where $\mathcal{I}$ encapsulates the integral variables, and matrix $A \in \mathbb{R}^{m \times n}$ has no special structure. The *linear relaxation* of $\mathcal{G}^R$ is

$$\mathcal{G}^R := \{Ax \geq b,\ x \geq 0,\ L_i \leq x_i \leq U_i\ \ \forall i \in I\} \tag{3}$$

where the integrality constraints are dropped. L_i, and U_i are the lower and upper bound of the i-th variable, respectively. The *branch-and-cut* heavily exploits this relaxation, thereby it is clear how the pre-requisite for tractable computations is an efficient *LP* solver.

[1] For an extended technical and historical review on *MIP*, we refer the reader to Lodi (2010), and Jünger (2009).

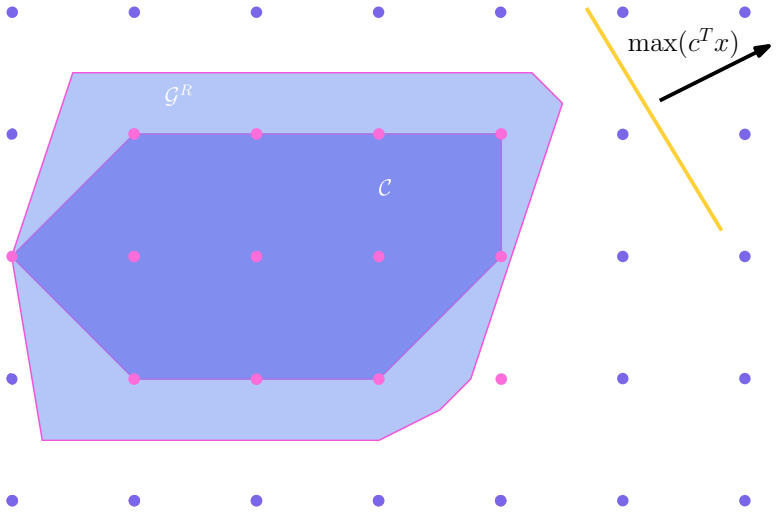

Figure 1: A *MIP* in the form of Equation (1).

On the hardness of *MIP* From a theoretical standpoint, we would like to optimize over the convex-hull of $\mathcal{G}$, or $\mathcal{C} := \text{conv}(\mathcal{G})$, so that any optimal solution fulfil the integrality requirements in $\mathcal{I}$. In other words, we aim for the *perfect formulation* of the set $\mathcal{G}$, so that the *MIP* can be solved as a linear program (Conforti et al., 2014). Such formulation is always guaranteed to exist when A, and b are rational, as proven by Meyer (1974). When there are only integer variables – or set $\mathcal{I}$ has cardinality n – then we refer to the *perfect formulation* as the *integral polyhedron*. However, these formulations can turn out to be computationally untractable, and require non-negligible computing resources. This is especially true when we model a $\mathcal{NP}$-hard problem with *MIP*s. Coming back to a general *MIP*, consider the feasibility problem associated with $\mathcal{G}$, asking to decide whether the set is empty or not. As a consequence of the result proven by Meyer (1974), we can always check a certificate in polynomial-time, by simply plugging it into the formulation. Hence, *MIP* is in $\mathcal{NP}$. Therefore, unless $\mathcal{P} = \mathcal{NP}$ (Garey and Johnson, 2009), these results suggest that computational matters within *MIP* are indeed fundamental to solve problems in practice.

1.1 Polyhedral combinatorics

A primer Polyhedral Combinatorics deals with the intrinsic geometrical properties of the polyhedron associated with $\mathcal{G}$. For the Minkowski-Weyl's

Algorithm 1 A generic *branch-and-cut* scheme

1: **Input:** The description of $\mathcal{G}$, and the vector c of objective coefficients.
2: **Output:** The optimal solution x^* or a certificate that none exist.
3: Initialize $\mathcal{L} := \{0\}$, and $z^* = -\infty$
4: **while** $\mathcal{L} \neq \emptyset$ **do**
5: Re-initialize $z_s = -\infty$, and $x_s = \bar{0}$
6: **Exploration Strategy**: pick a subproblem $S \in \mathcal{L}$, and set $\mathcal{L} = \mathcal{L} \backslash \{S\}$
7: **Solve LP**: solve the linear relaxation of the problem
8: **if** the relaxation has a feasible solution **then**
9: $z_s \leftarrow$ objective, and $x_s \leftarrow$ solution of S
10: **if** x_s fulfills integrality requirements *and* $z_s > z^*$ **then**
11: $z^* \leftarrow z_s$, and $x^* \leftarrow x_s$
12: **Fathoming**: delete from $\mathcal{L}$ nodes with upper bounds lower than z^*
13: **else**
14: **Cutting**: eventually, try to find valid cuts.
15: **if** x_s fulfills integrality requirements **then**
16: Go to Step 9
17: **end if**
18: **Branching**: Split S into k subproblems, $S = \bigcup_{i=1}^{k} S_i$, and update
19: $\mathcal{L} = \mathcal{L} \cup \bigcup_{i=1}^{k} S_i$
20: **end if**
21: **end if**
22: **end while**
23: **return** z^*, and x^*

theorem, a polyhedron can be described as the intersection of finitely many halfspaces. Alternatively, we can express it as a combination of extreme points and unbounded positive rays. The *dimension* of P – $\dim(P)$ – is the maximum number of affinely independent points in P minus 1. We define as *valid inequality* for P an inequality of the form $\pi^T x \geq \pi_0$ where $x \in P$. The face induced by a valid inequality $\pi^T x \geq \pi_0$ on P is $F(\pi, \pi_0) = \{x \in P : \pi^T x = \pi_0\}$, and it is also a polyhedron. If $\dim(F(\pi, \pi_0))$ is 0 we have a vertex of P, while if $\dim(F(\pi, \pi_0))$ is equal to $dim(P) - 1$ we have a *facet* of P. Hence, the natural question arising is which inequalities are necessary to give a *minimal description* of P. The answer is almost straightforward. The minimal description of P is given by the set of all its *facets*. Therefore, we can see that not all inequalities are as necessary as the facet-defining ones. In this spirit, *facet* inequalities are favored as *strong* compared to the other ones.

Example 1 (A valid inequality)**.** *Consider the set following set* $\mathcal{G}$

$$\max\{c^T x = x_1 + 10x_2 : x \in \mathcal{G}\} \tag{4}$$

$$\mathcal{G} = \{-3x_1 - 2x_2 \geq 3; x_1 \geq 0; x_2 \in \{0, 1\}\} \tag{5}$$

Solving the linear-relaxation of this problem yelds the solution $(x_1, x_2) = (0, \frac{3}{2})$*, and hence* x_2 *is not integral. It is easy to see that any cut* $\pi_1 x_1 + \pi_2 x_2 \geq \pi_0$ *is valid for* $\mathcal{G}$ *when* $\pi < 0$*,* $\pi_0 = -1$*, and* π_1/π_2 *is in* $[1, \frac{3}{2}]$*. In particular, by adding the cut* $\pi_0 = \pi_1 = -1$ *and solving again the linear relaxation, we retrieve optimal solution* $(x_1, x_2) = (0, 1)$*. Assume* $\mathcal{C} = conv(\mathcal{G})$*. Since* $dim(\mathcal{C}) = 2$*, and the valid inequality contains* 2 *points, then* $-x_1 - x_2 \geq -1$ *is a* facet *of* $\mathcal{C}$*.*

Seeking for perfection The central question is how to retrieve the *perfect-fomulation* of $\mathcal{C}$. This may or may not be the *minimal description* $\mathcal{C}$. As mentioned, this *perfect-formulation* is usually not genuinely easy to find, and hence may prevent any computational tractability. Hence, the objective is to capture an enhanced description of $\mathcal{G}^R$, which can hopefully yield an optimal (mixed) integer solution given a specific objective function $c^T x$. There is a straightforward practical procedure we can exploit to get this enhanced description, which can be summarized as follow. Start with R and solve its associated linear relaxation. If the solution $\bar{x}$ violates any of the integrality constraints, then derive a *valid* – and possibly strong – *inequality* to cut off the fractional point $\bar{x}$. In other words, we seek for an inequality in the form of $\pi^T x \geq \pi_0$ for all $x \in P$, and $\pi^T \bar{x} < \pi_0$. We iteratively solve the linear relaxation and find such inequalities until we obtain the optimal solution fulfilling the original integer requirements.

The separation problem The derivation of *valid inequalities* to strenghten the description of the relaxation roots in the so-called *separation problem* (Grötschel et al., 1981). Its name stems from the idea of separing $\bar{x}$ from $\mathcal{C}$ by finding a violated *valid inequality*. At each separation step, we obtain an enhanced description of $\mathcal{G}^R \leftarrow \mathcal{G}^R \cap \{\pi^T x \geq \pi_0\}$. A fundamental result concerning separation is the *equivalence between optimization and separation*. It states that the separation problem can be solved in polynomial-time only if the original *MIP* problem can be solved in polynomial time (Conforti et al., 2014). Informally, if an oracle gives us the best cut at each separation step for a $\mathcal{NP}$-hard problem, then we can solve the original problem in polynomial time. Most of the time, this is not the case, and the separation problem is $\mathcal{NP}$-hard itself. However, we hope the separation

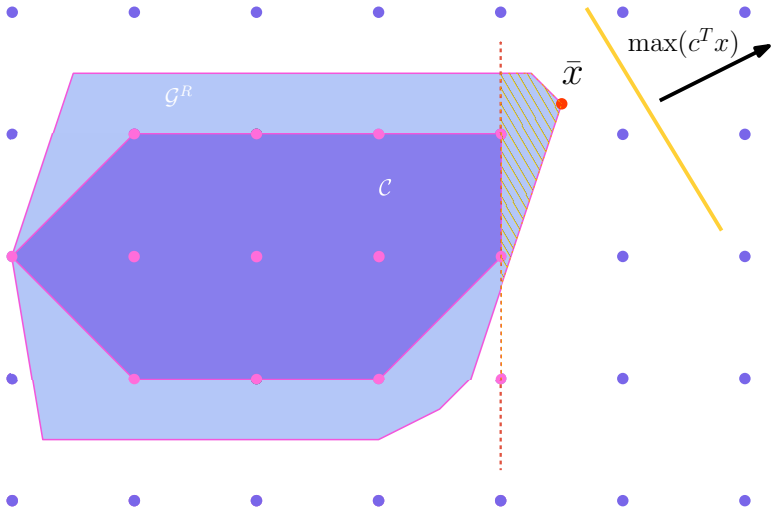

Figure 2: A valid inequality for the *MIP* in Figure 1. In this case, the inequality cuts off the point $\bar{x} \in \mathcal{G}^R \backslash \mathcal{C}$, and also defines a facet of $\mathcal{C}$.

problem can be practically solved more quickly than the original one. The interrogative is hence how the separation problem looks like, namely which types of *valid inequalities* we can separate. Generally speaking, we can separate either problem-specific or general-purpose inequalities.

Problem-specific cuts In the first case, the focal point is to find valid inequalities for well-defined paradigmatic Combinatorial Optimization problems, for instance, the Traveling Salesman Problem (Dantzig et al., 1954; Miller et al., 1960; Padberg and Rinaldi, 1990; Fischetti, 1991; Padberg and Rinaldi, 1991; Fischetti et al., 1995), the Knapsack Problem (Balas, 1975; Balas and Zemel, 1978; Gu et al., 1999), the Set Covering Problem (Balas and Ng, 1989). In this context, the synergies between Combinatorial Optimization, polyhedral combinatorics, and *MIP* provided a long-standing source of innovations. Many combinatorial optimization problems can be formulated as *MIP*s, and vice versa. Thereby, while exploiting problem-specific cuts, we implicitly assume A has a special structure to derive valid inequalities. A special note goes to Padberg and Rinaldi (1991), who developed the first efficient implementation of the so-called *branch-and-cut* framework for the Traveling Salesman Problem. The authors indeed separate a variety of valid inequalities for the problem, for instance, *subtour elimination* ones.

Example 2 (Knapsack Cover Cuts)**.** *Consider the set following knapsack*

problem with 4 items

$$\max\{c^T x = 22x_1 + 19x_2 + 17x_3 + 10x_4 : x \in \mathcal{G}\} \tag{6}$$

$$\mathcal{G} = \{12x_1 + 9x_2 + 6x_3 + 4x_4 \leq 15; x \in \{0,1\}^4\} \tag{7}$$

Intuitively, it is easy to see that picking x_2*,* x_3*, and* x_4 *yelds a capacity of* $19 \geq 15$*. Therefore, a valid inequality would enforce at most* 2 *of these elements are selected, namely* $x_2 + x_3 + x_4 \leq 2$*. This is called a* knapsack covering inequality*. We can strenghten (lift) it by adding item* x_1*, and alter its coefficient to* 2*. The facet defining inequality is then* $2x_1 + x_2 + x_3 + x_4 \leq 2$*.*

General-purpose cuts In the second case, we derive generally valid inequalities without exploiting any special structure of A. We refer to this class as general-purpose cutting planes. In this context, we remark the pioneering theoretical works of Chvátal (1973) and Gomory (1958) on the so-called *CG-cuts* for pure integer programs. The basic reasoning is to obtain a rounded down inequality from the known-one embedded in the description of A, namely $(\lambda A)x \leq \lfloor \lambda b \rfloor$. The term $\lambda \in \mathbb{R}^m_+$ is often called the vector of *CG*-multipliers. In general, such general-purpose cuts are generated by inducing disjunctive arguments on *Chvátal* inequalities (Lodi, 2010). A disjunction is a clause in the form of $A^1 x \geq b^1 \vee A^2 x \geq b^2 \vee \dots A^z x \geq b^z$ satisfied by all solutions in $\mathcal{C}$. Within this context, *Disjunctive Programming* (Balas, 2018) stemmed out to study the properties of disjunctions and their direct implications for *MIP* cutting-planes. Fischetti et al. (2011) provides some detailed computational consideration on the separation of general disjunctive cuts, for instance, the role of *normalization*. Remarkably, practically implemented disjunctive cuts are usually two-sided disjunctions, where namely there are only two disjunctive terms. In general, one can consider disjunction with more than two terms (Perregaard and Balas, 2001). An important class of general-purpose cuts is the one of *Mixed-Integer Gomory cuts* (*MIG*). Balas et al. (1996) pioneered their implementation in the late 90s. Nowadays, this is one of the most powerful classes of cuts implemented in modern solvers (Achterberg, 2009; Bixby et al., 2000). It is interesting to observe how the theoretical understanding anticipated – in terms of several decades – the implementations in modern *MIP* computations (Cornuéjols, 2007a). In fact, before the breakthrough work of Balas et al. (1996), general-purpose cutting planes were mostly considered because of their theoretical interest more than the practical one. Among the other families of cuts, we briefly cite:

(i) *Split cuts*, where given $c \in \mathbb{Z}^{|I|}$, $d \in \mathbb{Z}$, we derive a general disjunction by intersecting $\mathcal{G}^R$ either with $\sum_{i \in I} c_i x_i \leq d$ or $\sum_{i \in I} c_i x_i \geq d+1$ (Balas, 2018). Here the disjunction is one, and results in two different splits. *Mixed-Integer Gomory cuts*, and *lift-and-project*, and *Mixed-Integer Roundings* cuts are all splits.

(ii) $\{0, 1/2\}$-*CG* cuts, namely *CG-cuts* where all the coefficients of λ are either 0 or $1/2$. Caprara and Fischetti (1996) showed that such inequalities naturally arise in some formulations of combinatorial problems (e.g., the Traveling Salesman Problem).

(iii) *Lift-and-project cuts*. In their two-sided *binary* 0-1 form, they are split-cuts where the disjunction is made on a binary variable x_i, $i \in I$ so that $x_i \leq 0$ or $x_i \geq 1$ – namely where $\lambda \in \{0, 1\}$ for both sides (Zemel, 1978). The disjunction is then $\{Ax \geq b; x_i = 0\} \vee \{Ax \geq b; x_i = 0\}$. Their name derives from the idea of finding such cuts in a higher-dimensional polyhedron and projecting it back to the original space. In practice, this means solving a linear program with an enriched number of variables wrt the original problem. (Balas et al., 1993; Balas and Bonami, 2009). However, Balas and Perregaard (2003) proved that two-sided disjunctive lift-and-project cuts can be efficiently extracted from the simplex-tableau of the linear relaxation.

(iv) *Mixed Integer Rounding cuts* (MIR). The reasoning is similar to the one for MIG. Consider an inequality in the description of $\mathcal{G}$, and its a (a_i) vector and b (b_i) coefficient. Alternatively, we can also consider an aggregation of constraints. Let f_0 be $b - \lfloor b \rfloor$, and similarly f_j – for each variable $j \in [n]$ – be $a_j - \lfloor a_j \rfloor$. The rounding cut relies on the so-called rounding factor $1 - f_0$ (Nemhauser and Wolsey, 1988).

For a primer on their relationship we refer to Kazachkov (2018), and Balas and Perregaard (2003), while for a more didactic version to Conforti et al. (2014), and Cornuéjols (2007b). We remark also that – within a *branch-and-cut* framework – we need to distinguish on globally valid cuts (e.g., valid for any note in the search tree), or locally valid cuts (e.g., valid only for specific nodes). Indeed, we can either separate globally-valid inequalities or locally-valid ones.

1.2 Presolving and primal heuristics

Presolving Before the problem is fed to the *branch-and-cut* algorithm, a *MIP* solver usually performs the so-called *presolving*. In a nutshell, *presolving*

routines try to detect redundant information and seek to transform the given model into a possibly simpler one. A few basic routines are, for instance, coefficient reductions, empty or implied rows detection, and bound strengthening are just a few of these (Savelsbergh, 1994). Achterberg (2009) developed a set of conflict detection algorithms based on constraint programming. The thesis showcases both general presolving algorithms and specialized routines for a class of constraints (e.g., knapsack constraint preprocessing). The extent of sophistication of *presolving* techniques plays a fundamental role in modern *MIP* solvers. For a more detailed review on the topic, we refer to Achterberg (2009), and Achterberg et al. (2019).

Example 3 (Basic Presolving)**.** *Consider the set following set* $\mathcal{G}$

$$\begin{aligned}\mathcal{G} = \{&1200x_1 + 900x_2 + 300x_3 \leq 600;\\ &x_3 \geq 1;\ x_1 + x_2 \leq 1;\ x \in \{0,1\}^3\}\end{aligned} \tag{8}$$

The coefficients in the first inequality can be reduced *to* $4x_1 + 3x_2 + 1x_3 \leq 2$*. Since the second inequality forces* x_3 *to be at its upper bound, then* $x_3 = 1$*, and the first inequality reads as* $4x_1 + 3x_2 \leq 1$*. Furthermore, this inequality is* implied *from the third one.*

Primal heuristics The *branch-and-cut* algorithm is guaranteed to terminate in a finite amount of time and return either a solution or a certificate of infeasibility. Furthermore, the *fathoming* step of the algorithm strongly relies on the quality of the incumbent solution, and hence the better incumbent the more the pruning. However, Achterberg (2009) noted that heuristics cannot significantly speed up optimality proving within solvers. Besides, quickly finding a feasible solution is often fundamental for the end-user. *Primal heuristics* are algorithms seeking for feasible solutions, possibly qualitatively good ones, within a short execution time. Despite they not guaranteed to retrieve a solution, they are a fundamental ingredient of modern *MIP* computations (Berthold, 2014). According to Fischetti and Lodi (2011), we can distinguish mainly three categories of primal heuristics:

(i) *Rounding Heuristics* starts often from an *LP* solution violating some integrality requirements. The heuristic tries then to *round* the violated components, for instance to the nearest integer value. We refer to (Achterberg, 2009) for an extended on different rounding techniques.

(ii) *Diving Heuristics* fix an integer variable to some value starting from the LP solution. The value and direction of this bounding are given by the

diving strategy. Fractional, pseudo-cost, line-search divings are a few examples. A notorious heuristic in this class is the *Feasability Pump* (Fischetti et al., 2005), where the diving is performed by altering the objective function of the original problem. Specifically, this heuristic builds two sequences of points that may eventually converge to a feasible solution (Berthold et al., 2019). The first sequence is made of *LP* feasible points, while the second is made of integer points – possibly infeasible for the *LP* relaxation.

(iii) *Improving Heuristics* try to build a better solution starting from a feasible one. These heuristics often solve smaller *MIP*s to search within a neighborhood of the incumbent solution. *Local Branching* – introduced by Fischetti and Lodi (2003) – branches on the hamming distance with respect to the incumbent solution.

A final note goes to the *MIPping* approach, introduced Fischetti et al. (2009). In a nutshell, the *MIP* solver is conceived as a black-box, and fundamental decisions within the *branch-and-cut* are transformed into *MIP*s. Therefore, the solver leverages on itself to formulate smaller (and hopefully easier to solve) subproblems to rescue the solution of the original problem.

1.3 Branching

Not all separations are equal The branching process involves two separate decisions. First, the *node selection* to determine the next node to be processed. Secondly, the *variable selection* problem determines which variable to branch on. Note that the problem S at a given node can be split into k subproblems, $S = \bigcup_{i=1}^{k} S_i$. However, modern *MIP* solvers, branching on general inequalities is seldom (Achterberg, 2009).

Node selection The two most popular rules are *best first*, and *depth first* (Achterberg et al., 2005; Lodi, 2010). The first one explores the nodes promising the best bound – the one with the highest lower bound in a maximization problem. Despite this rule seeks for the most promising solution, it might be computationally intensive since it jumps between possibly different parts of the search tree. Indeed, the *depth first* strategy reoptimizes – usually with the dual-simplex – the same *LP* with an added constraint due to the branching.

Variable selection Instead of branching on general inequalities, a more common choice is to branch on a specific variable. In particular, a fractional

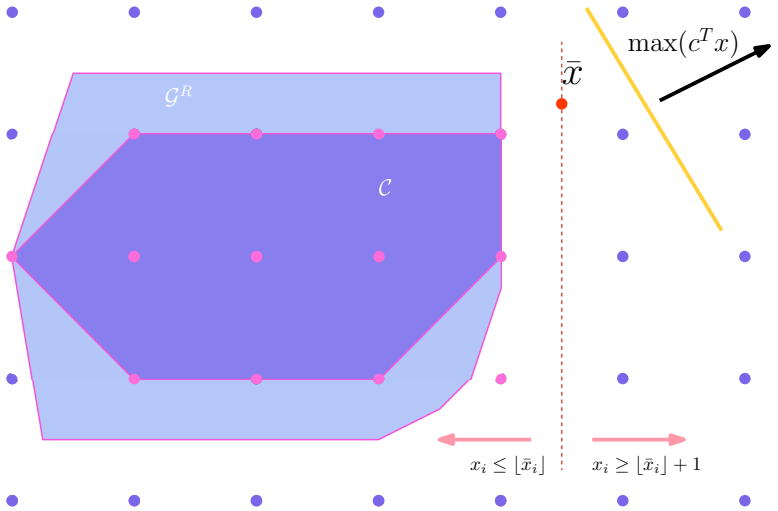

Figure 3: Branching on the variable on the horizontal axis. Note that we can see branching as a general disjunctive argument. Indeed, the branching decision defines a facet of $\mathcal{C}$, as in Figure 2.

variable x_i with $i \in I$ violating the integral requirements in the solution of the *LP* relaxation. Thereby, we create two subproblems with the disjunction $x_i \leq \lfloor \bar{x}_i \rfloor$ or $x_i \geq \lfloor \bar{x}_i \rfloor + 1$. The two *dichotomical* subproblems are usually referred as *left* – or *down* – branch, and *right* – or *up* – branch, respectively. For a pictorial representation, see Figure 3. The key question is how to determine the variable to branch on. We can sum up the most common branching rules as follow:

(i) The *most infeasible brancing* selects as branching variable the one closest to 0.5 – namely the farthest from being integral. The *least infeasible brancing* is dual to the latter one, and selects as branching variable the one closest from being integer. Achterberg et al. (2005) shows that these rules both perform no better than a random choice.

(ii) *Pseudocosts* branching (Benichou et al., 1971) try to estimate the per-unit decrease of the lower bound associated to each variable x_i. In specific, we associate two scores P_i^+, and P_i^-, to estimate the per-unit decrease in the bound. Let the fractional variable x_i take value $\bar{x}_i = \lfloor \bar{x}_i \rfloor + f_i$, where $f_i \geq 0$ is its fractional part. Then, at a node j, $D_i^{j-} = P_i^- f_i$, and $D_i^{j+} = P_i^+(1 - f_i)$ are the *down* branch and *up* branch estimates, respectively. We remark P_i^+, P_i^- are updated as

soon as the *branch-and-cut* branches on variable i. Hence, these two statistics are not available early in the search tree.

(iii) *Strong branching* computes – instead of estimating – the impact on the bound for each branching candidate (Applegate et al., 1995, 2006). Hence, it solves two LPs for each of the latter.

(iv) *Reliability pseudocost branching* hybridize *strong* and *pseudocosts* branching. The rule triggers *strong branching* on variables with uninitialized pseudocosts. Furthermore, it sets a *reliability* threshold on the minimum value of the pseudocosts associated to each variable (Achterberg et al., 2005).

(v) *Hybrid branching* combines five different scores taken from *MIP* and constraint satisfaction contexts (Achterberg and Berthold, 2009)

As remarked by Lodi (2010), *branching* decisions – intrinsically heuristic in their nature – are indeed a crucial component of the *MIP* technology.

1.4 Machine learning and MIP

ML tasks usually involve the estimation of some unknown parameter from raw – and possibly noisy – data. A recent survey on *CO* and *ML* from Bengio et al. (2018) highlights some methodological characteristics of the interaction between *CO* and *ML*. First, one may seek for an *ML* algorithm to predict the final solution of a combinatorial problem. In this case, the learning process exploits the peculiar structure of the problem. This approach is then of less interest to the *MIP* framework because of its constrained generalization capabilities. However, it is still amusing for specific problems. For instance, Dai et al. (2018) tries to learn greedy algorithms for three combinatorial problems. Larsen et al. (2018) proposes a novel way to learn tactical solutions to planning problems under uncertainty. Thereby, *ML* deals with the stochasticity due to the incomplete information. Another option is to exploit *ML* to learn algorithmic configurations. Iommazzo (2019) selects the solver configuration for a given instance using *ML*. Bonami et al. (2018) learn whether to linearize or not the objective function of Mixed-Integer Quadratic Programming Problems in $CPLEX$. In a different context, Fischetti et al. (2019) seeks to predict whether a *MIP* will be solved to optimality before reaching a given time limit. As noted by Lodi and Zarpellon (2017) – these approaches are more suitable for integrations with a general *MIP* framework, since they generalize beyond *CO* and specific problems. In particular, several

authors worked within the integration of *branching* and *ML*. Gasse et al. (2019) recently proposed a graph convolutional neural network to learn *variable selection* policies within a branching scheme. Khalil et al. (2016) propose a support vector machine ranking system to imitate *strong branching*. For a full survey on learning and branching, we refer to Lodi and Zarpellon (2017). To conclude, because of the heuristic nature of *MIP* computations, the scope of integration of *ML* and *MIP* is broad and appealing for future research.

References

Achterberg, T., 2009. Constraint Integer Programming. Ph.D. thesis. Technischen Universität Berlin.

Achterberg, T., Berthold, T., 2009. Hybrid Branching, in: van Hoeve, W.J., Hooker, J.N. (Eds.), Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. Springer Berlin Heidelberg, Berlin, Heidelberg. volume 5547, pp. 309–311. URL: http://link.springer.com/10.1007/978-3-642-01929-6_23, doi:10.1007/978-3-642-01929-6_23.

Achterberg, T., Bixby, R.E., Gu, Z., Rothberg, E., Weninger, D., 2019. Presolve Reductions in Mixed Integer Programming. INFORMS Journal on Computing , ijoc.2018.0857URL: http://pubsonline.informs.org/doi/10.1287/ijoc.2018.0857, doi:10.1287/ijoc.2018.0857.

Achterberg, T., Koch, T., Martin, A., 2005. Branching rules revisited. Operations Research Letters 33, 42–54. URL: https://linkinghub.elsevier.com/retrieve/pii/S0167637704000501, doi:10.1016/j.orl.2004.04.002.

Applegate, D., Bixby, R., Chvátal, V., Cook, W., 1995. Finding cuts in the TSP (A preliminary report). volume 95. Citeseer.

Applegate, D.L., Bixby, R.E., Chvatal, V., Cook, W.J., 2006. The traveling salesman problem: a computational study. Princeton university press.

Balas, E., 1975. Facets of the knapsack polytope. Mathematical Programming 8, 146–164. URL: http://link.springer.com/10.1007/BF01580440, doi:10.1007/BF01580440.

Balas, E., 2018. Disjunctive programming. Springer Berlin Heidelberg, New York, NY.

Balas, E., Bonami, P., 2009. Generating lift-and-project cuts from the LP simplex tableau: open source implementation and testing of new variants. Math. Prog. Comp. 1, 165–199. URL: http://link.springer.com/10.1007/s12532-009-0006-4, doi:10.1007/s12532-009-0006-4.

Balas, E., Ceria, S., Cornuéjols, G., 1993. A lift-and-project cutting plane algorithm for mixed 0–1 programs. Mathematical Programming 58, 295–324. URL: http://link.springer.com/10.1007/BF01581273, doi:10.1007/BF01581273.

Balas, E., Ceria, S., Cornuéjols, G., Natraj, N., 1996. Gomory cuts revisited. Operations Research Letters 19, 1–9. URL: https://linkinghub.elsevier.com/retrieve/pii/0167637796000077, doi:10.1016/0167-6377(96)00007-7.

Balas, E., Ng, S.M., 1989. On the set covering polytope: I. All the facets with coefficients in {0, 1, 2}. Mathematical Programming 43, 57–69. URL: http://link.springer.com/10.1007/BF01582278, doi:10.1007/BF01582278.

Balas, E., Perregaard, M., 2003. A precise correspondence between lift-and-project cuts, simple disjunctive cuts, and mixed integer gomory cuts for 0-1 programming. Mathematical Programming 94, 221–245. URL: http://link.springer.com/10.1007/s10107-002-0317-y, doi:10.1007/s10107-002-0317-y.

Balas, E., Zemel, E., 1978. Facets of the Knapsack Polytope From Minimal Covers. SIAM Journal on Applied Mathematics 34, 119–148. URL: http://epubs.siam.org/doi/10.1137/0134010, doi:10.1137/0134010.

Bengio, Y., Lodi, A., Prouvost, A., 2018. Machine Learning for Combinatorial Optimization: a Methodological Tour d'Horizon. arXiv:1811.06128 [cs, stat] URL: http://arxiv.org/abs/1811.06128. arXiv: 1811.06128.

Benichou, M., Gauthier, J.M., Girodet, P., Hentges, G., Ribiere, G., Vincent, O., 1971. Experiments in mixed-integer linear programming. Mathematical Programming 1, 76–94. URL: http://link.springer.com/10.1007/BF01584074, doi:10.1007/BF01584074.

Berthold, T., 2014. Heuristic algorithms in global MINLP solvers. Ph.D. thesis. Technischen Universität Berlin. URL: http://www.zib.de/berthold/Berthold2014.pdf.

Berthold, T., Lodi, A., Salvagnin, D., 2019. Ten years of feasibility pump, and counting. EURO J Comput Optim 7, 1–14. URL: http://link.springer.com/10.1007/s13675-018-0109-7, doi:10.1007/s13675-018-0109-7.

Bixby, E.R., Fenelon, M., Gu, Z., Rothberg, E., Wunderling, R., 2000. MIP: Theory and Practice — Closing the Gap, in: Powell, M.J.D., Scholtes, S. (Eds.), System Modelling and Optimization. Springer US, Boston, MA. volume 46, pp. 19–49. URL: http://link.springer.com/10.1007/978-0-387-35514-6_2, doi:10.1007/978-0-387-35514-6_2.

Bonami, P., Lodi, A., Zarpellon, G., 2018. Learning a Classification of Mixed-Integer Quadratic Programming Problems, in: van Hoeve, W.J. (Ed.), Integration of Constraint Programming, Artificial Intelligence, and Operations Research. Springer International Publishing, Cham. volume 10848, pp. 595–604. URL: http://link.springer.com/10.1007/978-3-319-93031-2_43, doi:10.1007/978-3-319-93031-2_43.

Caprara, A., Fischetti, M., 1996. {0, 1/2}-Chvátal-Gomory cuts. Mathematical Programming 74, 221–235. URL: http://link.springer.com/10.1007/BF02592196, doi:10.1007/BF02592196.

Chvátal, V., 1973. Edmonds polytopes and a hierarchy of combinatorial problems. Discrete Mathematics 4, 305–337. URL: https://linkinghub.elsevier.com/retrieve/pii/0012365X73901672, doi:10.1016/0012-365X(73)90167-2.

Conforti, M., Cornuéjols, G., Zambelli, G., 2014. Integer Programming. volume 271 of *Graduate Texts in Mathematics*. Springer International Publishing, Cham. URL: http://link.springer.com/10.1007/978-3-319-11008-0, doi:10.1007/978-3-319-11008-0.

Cornuéjols, G., 2007a. Revival of the Gomory cuts in the 1990's. Ann Oper Res 149, 63–66. URL: http://link.springer.com/10.1007/s10479-006-0100-1, doi:10.1007/s10479-006-0100-1.

Cornuéjols, G., 2007b. Valid inequalities for mixed integer linear programs. Mathematical Programming 112, 3–44. URL: http://link.springer.com/10.1007/s10107-006-0086-0, doi:10.1007/s10107-006-0086-0.

Dai, H., Khalil, E.B., Zhang, Y., Dilkina, B., Song, L., 2018. Learning Combinatorial Optimization Algorithms over Graphs. arXiv:1704.01665 [cs, stat] URL: http://arxiv.org/abs/1704.01665. arXiv: 1704.01665.

Dantzig, G., Fulkerson, R., Johnson, S., 1954. Solution of a Large-Scale Traveling-Salesman Problem. Journal of the Operations Research Society of America 2, 393–410. URL: http://pubsonline.informs.org/doi/abs/10.1287/opre.2.4.393, doi:10.1287/opre.2.4.393.

Fischetti, M., 1991. Facets of the Asymmetric Traveling Salesman Polytope. Mathematics of Operations Research 16, 42–56. URL: http://pubsonline.informs.org/doi/abs/10.1287/moor.16.1.42, doi:10.1287/moor.16.1.42.

Fischetti, M., Glover, F., Lodi, A., 2005. The feasibility pump. Math. Program. 104, 91–104. URL: http://link.springer.com/10.1007/s10107-004-0570-3, doi:10.1007/s10107-004-0570-3.

Fischetti, M., González, J.J.S., Toth, P., 1995. The symmetric generalized traveling salesman polytope. Networks 26, 113–123. URL: http://doi.wiley.com/10.1002/net.3230260206, doi:10.1002/net.3230260206.

Fischetti, M., Lodi, A., 2003. Local branching. Mathematical Programming 98, 23–47. URL: http://link.springer.com/10.1007/s10107-003-0395-5, doi:10.1007/s10107-003-0395-5.

Fischetti, M., Lodi, A., 2011. Heuristics in Mixed Integer Programming, in: Wiley Encyclopedia of Operations Research and Management Science. John Wiley & Sons, Inc., Hoboken, NJ, USA, p. eorms0376. URL: http://doi.wiley.com/10.1002/9780470400531.eorms0376, doi:10.1002/9780470400531.eorms0376.

Fischetti, M., Lodi, A., Salvagnin, D., 2009. Just MIP it!, in: Maniezzo, V., Stützle, T., Voß, S. (Eds.), Matheuristics. Springer US, Boston, MA. volume 10, pp. 39–70. URL: http://link.springer.com/10.1007/978-1-4419-1306-7_2, doi:10.1007/978-1-4419-1306-7_2.

Fischetti, M., Lodi, A., Tramontani, A., 2011. On the separation of disjunctive cuts. Math. Program. 128, 205–230. URL: http://link.springer.com/10.1007/s10107-009-0300-y, doi:10.1007/s10107-009-0300-y.

Fischetti, M., Lodi, A., Zarpellon, G., 2019. Learning MILP Resolution Outcomes Before Reaching Time-Limit, in: Rousseau, L.M., Stergiou, K. (Eds.), Integration of Constraint Programming, Artificial Intelligence, and Operations Research. Springer International Publishing, Cham. volume 11494, pp. 275–291. URL: http://link.springer.com/10.1007/978-3-030-19212-9_18, doi:10.1007/978-3-030-19212-9_18.

Garey, M.R., Johnson, D.S., 2009. Computers and intractability: a guide to the theory of NP-completeness. A series of books in the mathematical sciences. 27. print ed., Freeman, New York [u.a]. OCLC: 551912424.

Gasse, M., Chetelat, D., Ferroni, N., Charlin, L., Lodi, A., 2019. Exact Combinatorial Optimization with Graph Convolutional Neural Networks. Proceding of NIPS 2019 , 13.

Gomory, R.E., 1958. Outline of an algorithm for integer solutions to linear programs. Bull. Amer. Math. Soc. 64, 275–279. URL: http://www.ams.org/journal-getitem?pii=S0002-9904-1958-10224-4, doi:10.1090/S0002-9904-1958-10224-4.

Grötschel, M., Lovász, L., Schrijver, A., 1981. The ellipsoid method and its consequences in combinatorial optimization. Combinatorica 1, 169–197. URL: http://link.springer.com/10.1007/BF02579273, doi:10.1007/BF02579273.

Gu, Z., Nemhauser, G.L., Savelsbergh, M.W.P., 1999. Lifted Cover Inequalities for 0-1 Integer Programs: Complexity. INFORMS Journal on Computing 11, 117–123. URL: http://pubsonline.informs.org/doi/abs/10.1287/ijoc.11.1.117, doi:10.1287/ijoc.11.1.117.

Iommazzo, G., 2019. Algorithmic configuration by learning and optimization.

Jünger, M. (Ed.), 2009. 50 years of integer programming, 1958-2008: the early years and state-of-the-art surveys. Springer, Heidelberg.

Kazachkov, A.M., 2018. Non-recursive cut generation. Ph.D. thesis. Carnegie Mellon University. URL: https://kilthub.cmu.edu/ndownloader/files/12255308.

Khalil, E.B., Bodic, P.L., Song, L., Nemhauser, G., Dilkina, B., 2016. Learning to Branch in Mixed Integer Programming. Proceding of NIPS 2019 , 8.

Land, A.H., Doig, A.G., 1960. An Automatic Method of Solving Discrete Programming Problems. Econometrica 28, 497. URL: https://www.jstor.org/stable/1910129?origin=crossref, doi:10.2307/1910129.

Larsen, E., Lachapelle, S., Bengio, Y., Frejinger, E., Lacoste-Julien, S., Lodi, A., 2018. Predicting Tactical Solutions to Operational Planning Problems under Imperfect Information. arXiv:1807.11876 [cs, stat] URL: http://arxiv.org/abs/1807.11876. arXiv: 1807.11876.

Lodi, A., 2010. Mixed Integer Programming Computation, in: Jünger, M., Liebling, T.M., Naddef, D., Nemhauser, G.L., Pulleyblank, W.R., Reinelt, G., Rinaldi, G., Wolsey, L.A. (Eds.), 50 Years of Integer Programming 1958-2008. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 619–645. URL: http://link.springer.com/10.1007/978-3-540-68279-0_16, doi:10.1007/978-3-540-68279-0_16.

Lodi, A., Zarpellon, G., 2017. On learning and branching: a survey. TOP 25, 207–236. URL: http://link.springer.com/10.1007/s11750-017-0451-6, doi:10.1007/s11750-017-0451-6.

Meyer, R.R., 1974. On the existence of optimal solutions to integer and mixed-integer programming problems. Mathematical Programming 7, 223–235. URL: http://link.springer.com/10.1007/BF01585518, doi:10.1007/BF01585518.

Miller, C.E., Tucker, A.W., Zemlin, R.A., 1960. Integer Programming Formulation of Traveling Salesman Problems. Journal of the ACM (JACM) 7, 326–329. URL: http://dl.acm.org/doi/10.1145/321043.321046, doi:10.1145/321043.321046.

Nemhauser, G., Wolsey, L., 1988. Integer and Combinatorial Optimization: Nemhauser/Integer and Combinatorial Optimization. John Wiley & Sons, Inc., Hoboken, NJ, USA. URL: http://doi.wiley.com/10.1002/9781118627372, doi:10.1002/9781118627372.

Padberg, M., Rinaldi, G., 1990. Facet identification for the symmetric traveling salesman polytope. Mathematical Programming 47, 219–257. URL: http://link.springer.com/10.1007/BF01580861, doi:10.1007/BF01580861.

Padberg, M., Rinaldi, G., 1991. A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems. SIAM Review 33, 60–100. URL: http://www.jstor.org/stable/2030652.

Perregaard, M., Balas, E., 2001. Generating Cuts from Multiple-Term Disjunctions, in: Goos, G., Hartmanis, J., van Leeuwen, J., Aardal, K., Gerards, B. (Eds.), Integer Programming and Combinatorial Optimization. Springer Berlin Heidelberg, Berlin, Heidelberg. volume 2081, pp. 348–360. URL: http://link.springer.com/10.1007/3-540-45535-3_27, doi:10.1007/3-540-45535-3_27. series Title: Lecture Notes in Computer Science.

Savelsbergh, M.W.P., 1994. Preprocessing and Probing Techniques for Mixed Integer Programming Problems. ORSA Journal on Computing 6, 445–454. URL: http://pubsonline.informs.org/doi/abs/10.1287/ijoc.6.4.445, doi:10.1287/ijoc.6.4.445.

Zemel, E., 1978. Lifting the facets of zero–one polytopes. Mathematical Programming 15, 268–277. URL: http://link.springer.com/10.1007/BF01609032, doi:10.1007/BF01609032.