Continuous cutting plane algorithms in integer programming

Didier Chételat\textsuperscript{a}, Andrea Lodi\textsuperscript{b}

\textsuperscript{a}CERC, Polytechnique Montréal, Montréal, Canada
\textsuperscript{b}Cornell Tech and Technion – IIT, New York, USA

Abstract

Cutting planes for mixed-integer linear programs (MILPs) are typically computed in rounds by iteratively solving optimization problems, the so-called separation. Instead, we reframe the problem of finding good cutting planes as a continuous optimization problem over weights parametrizing families of valid inequalities. This problem can also be interpreted as optimizing a neural network to solve an optimization problem over subadditive functions, which we call the subadditive primal problem of the MILP. To do so, we propose a concrete two-step algorithm, and demonstrate empirical gains when optimizing generalized Gomory mixed-integer inequalities over various classes of MILPs. Code for reproducing the experiments can be found at https://github.com/dchetelat/subadditive.

1. Introduction

A mixed-integer linear programming (MILP) problem is an optimization problem of the form

\[
\begin{align*}
\min_{(x,z)\in \mathbb{R}^n} & \quad c^t x + h^t z \\
\text{s.t.} & \quad Ax + Gz \geq b, \\
& \quad x, z \geq 0, \quad x \in \mathbb{Z}^k,
\end{align*}
\]

(1)

where \(A \in \mathbb{R}^{m \times k}\), \(G \in \mathbb{R}^{m \times (n-k)}\), \(c \in \mathbb{R}^k\), \(h \in \mathbb{R}^{n-k}\) and \(b \in \mathbb{R}^m\). To solve such class of optimization problems, it is often useful to find lower (dual) bounds, i.e., values that optimistically approximate the MILP optimal value. Besides being useful to conservatively estimate how far from optimality a feasible solution might be, dual bounds are also useful in the branch-and-bound algorithm that forms the backbone of current state-of-the-art integer programming solvers \([I]\). The standard approach to obtain a dual bound is by solving the linear programming (LP) relaxation

\[
\begin{align*}
\min_{(x,z)\in \mathbb{R}^n} & \quad c^t x + h^t z \\
\text{s.t.} & \quad Ax + Gz \geq b, \\
& \quad x, z \geq 0
\end{align*}
\]

(2)

to the original MILP \([I]\), i.e., the problem obtained by dropping the integrality requirement on the \(x\) variables. For simplicity, we will assume both problems are feasible and bounded, so have optimal solutions.

A valid inequality for the MILP is any linear inequality \(\alpha^t x + \gamma^t z \geq \beta\) that is satisfied by its feasible set, i.e., such that \(\alpha^t x + \gamma^t z \geq \beta \ \forall (x, z) \in \{Ax + Gz \geq b, (x, z) \geq 0, x \in \mathbb{Z}^k\}\). If furthermore there exists a point in the feasible set of the LP relaxation \([2]\) that is not satisfied by this inequality, i.e., if there exists an \((\bar{x}, \bar{z}) \in \{Ax + Gz \geq b, (x, z) \geq 0\}\) such that \(\alpha^t \bar{x} + \gamma^t \bar{z} < \beta\), the inequality is called a cutting plane (or cut).

Algorithms exist that, given a MILP, produce cutting planes “by construction”. The resulting cutting planes are useful because the MILP with the cuts added is equivalent to the original MILP, but has a tighter linear relaxation, hence with equal or better dual bound. Usually, such algorithms proceed by solving a relaxation of the original MILP, most commonly the LP relaxation, and from its solution a collection of cutting planes, called a round, is produced. These cuts are added to the original MILP and the process is repeated again on this extended MILP, yielding a second round of cuts that can be added again to the MILP. This is repeated until convergence, or some other stopping criterion, to form a so-called cutting plane algorithm (see, e.g., \([I]\)).
A major advantage of this approach is that monotone improvement is guaranteed, in that adding more cuts cannot worsen the dual bound. However, at the same time, as the number of cuts added grows, at every round the intermediate optimization problems grow in size, leading to increased computational cost, diminishing returns and reduced numerical stability.

In this paper, we propose an alternative approach to computing good cuts. Instead of devising an algorithm that produces good cuts by construction, we propose to consider a large parametrized family of inequalities that are valid by construction, but search among them for the best cuts. Our proposed algorithm alternates between solving the LP relaxation of the problem with the tentative cuts added, and taking gradient steps onto the inequality parameters so as to cut off the best solution found so far. Unlike a classical approach, the number of cuts remains fixed throughout the optimization, and the tentative cuts can be interpreted as progressively deformed or replaced in an attempt to distill an effective subset. Experiments show that this approach can find cuts leading to better dual bounds than the classical approach, although the computational cost of this search does not, for the moment, make it competitive compared with these classical methods. However, it offers a novel approach to computing cuts that completely circumvents issues regarding the increase in LP size, diminishing returns or numerical stability.

The paper is organized as follows. In Section 2 we introduce subadditivity as a way of generating valid inequalities for MILP and we recall the interpretation of Gomory mixed-integer cuts as subadditive inequalities. In Section 3 we generalize the Gomory construction, while in Section 4 we formulate the optimization problem at the core of our separation algorithm. In Section 5 we describe an interesting connection where both the classical GMI algorithm and our proposed approach can be interpreted as the training of a subadditive neural network, highlighting potential connections to machine learning. Our algorithm is presented in Section 6 and its performance is computationally evaluated in Section 7. Finally, we conclude in Section 8.

In the remainder of the paper, we will use the following notation. Inequality between two vectors \( x, y \in \mathbb{R}^m \), denoted \( x \leq y \), is taken to mean component-wise inequality, i.e., \( x_i \leq y_i \) for all \( 1 \leq i \leq m \). We also assume that functions apply to matrices column-wise: that is, for a function \( f : \mathbb{R}^m \rightarrow \mathbb{R}^p \), and a matrix \( X = [X_1, \ldots, X_n] \in \mathbb{R}^{mxn} \) with columns \( X_1, \ldots, X_n \), we write \( f(X) \) for the \( p \times n \) matrix \( [f(X_1), \ldots, f(X_n)] \) with columns \( f(X_1), \ldots, f(X_n) \). A diagonal matrix with diagonal vector \( v \) will be denoted \( \Delta(v) \). Let us denote \( \mathbb{R} = \mathbb{R} \cup \{-\infty, +\infty\} \) the extended real line, and \( \lfloor x \rfloor = x - \{x\} \) the fractional part function. The composition of two functions \( f, g \) will be denoted \( f \circ g \), and the concatenation of two vectors \( x \in \mathbb{R}^m, x \in \mathbb{R}^m \) will be denoted \( [x, y] \in \mathbb{R}^{m+m'} \).

2. Subadditive valid inequalities

One way of generating valid inequalities is through non-decreasing, centered, subadditive functions. A function \( \phi(y) : \mathbb{R}^m \rightarrow \mathbb{R}^p \) is subadditive if \( \phi(y_1 + y_2) \leq \phi(y_1) + \phi(y_2) \) for any \( y_1, y_2 \). We furthermore assume that \( \phi \) is non-decreasing, that is \( \phi(y_1) \leq \phi(y_2) \) for any \( y_1 \leq y_2 \), and centered, that is \( \phi(0) = 0 \). Let us denote \( \mathcal{F}^{m,p} \) the set of all such functions.

Associated with any centered subadditive function \( \phi \) is another function, its Upper Directional Derivative at Zero (UDDZ) \( \bar{\phi} : \mathbb{R}^m \rightarrow \mathbb{R}^p \), defined by

\[
\bar{\phi}(y) = \lim \sup_{h \downarrow 0} \frac{\phi(hy)}{h}.
\]

This function is sublinear, i.e., it is subadditive and satisfies \( \phi(\lambda y) = \lambda \phi(y) \) for any \( y \in \mathbb{R}^m \) and \( \lambda \geq 0 \). Furthermore, it upper bounds \( \phi \), that is \( \phi(y) \leq \phi \bar{\phi}(y) \) for any \( y \in \mathbb{R}^m \).

Subadditivity is closely connected to cutting planes in the following way. For a function \( \phi \in \mathcal{F}^{m,p} \), the \( p \) inequalities

\[
\phi(A)x + \bar{\phi}(G)z \geq \phi(b)
\]

are always valid for the MILP \([1]\) (see, e.g., [2]). That is, a subadditive function can be used as a cut-generating function.

One example is provided by Gomory mixed-integer (GMI) inequalities \([3]\). Let \( B \) denote the \( m \times m \) optimal basis matrix of the LP relaxation \([2]\) in equality form, that is, the matrix composed of the
columns of $[A, G, -I]$ in the LP optimal basis. Then, the GMI inequalities are the $m$ cutting planes induced by the function $\phi \in \mathcal{F}^{m, m}$ defined by
\[
\phi(y) = \min \left\{ (B^{-1}y), \Delta \left( \frac{[B^{-1}]}{1 - [B^{-1}]} \right) (1 - [B^{-1}y]) \right\} \\
+ \max \left\{ -B^{-1}, \Delta \left( \frac{[B^{-1}]}{1 - [B^{-1}]} \right) B^{-1}y, \right\},
\]
(see, e.g., [4, Eq. (8)-(9)]) whose UDDZ is necessarily cut off by the inequalities. Indeed, Gomory [3] showed that we could have written the enlarged MILP (3) more tightly by considering a parametrized family of subadditive functions.

We note that if we define the function $f(y) = [y, \phi(y)]$, where we recall that $[\cdot, \cdot]$ denotes the concatenation of two vectors, then $\bar{f}(y) = [y, \tilde{\phi}(y)]$, and we could have written the enlarged MILP (3) more suggestively as
\[
\min_{(x,z) \in \mathbb{R}^n} \ c'x + h'z \\
\text{s.t.} \ Ax + Gz \geq b, \\
\phi(A)x + \tilde{\phi}(G)z \geq \phi(b), \\
x, z \geq 0, \ x \in \mathbb{Z}^k.
\]
which is equivalent to the original one (1), but has tighter LP relaxation, hence would hopefully yield a better dual bound. Indeed, Gomory [3] showed that the inequalities necessarily cut off the optimal LP solution i.e., they are cutting planes.

We note that if we define the function $g(y) = y$, where we recall that $\cdot$ denotes the concatenation of two vectors, then $\bar{f}(y) = [y, \tilde{\phi}(y)]$, and we could have written the enlarged MILP (3) more suggestively as
\[
\min_{(x,z) \in \mathbb{R}^n} \ c'x + h'z \\
\text{s.t.} \ f(A)x + \tilde{f}(G)z \geq f(b), \\
x, z \geq 0, \ x \in \mathbb{Z}^k.
\]
This is a MILP in the same form as the original problem (1), but with $A, G$ and $b$ replaced by $f(A), \tilde{f}(G)$ and $f(b)$ respectively.

3. Generalized GMI valid inequalities

Our reasoning starts by noting that we could have generalized GMIIs by instead considering a parametrized family of subadditive functions. Namely, we could have considered the functions
\[
\phi_{W,v}(y) = \min \left\{ \{W y\}, \Delta \left( \frac{v}{1 - v} \right) (1 - \{W y\}) \right\} \\
+ \max \left\{ -W, \Delta \left( \frac{v}{1 - v} \right) W \right\} y
\]
parametrized by an arbitrary weight matrix $W \in \mathbb{R}^{m \times m}$ and a vector $v \in [0, 1)^m$. One can recover the original function by setting $m' = m, W = B^{-1}$ and $v = \{B^{-1}b\}$.

(We remark in passing that another family of generalized GMI inequalities has also been considered in the literature: a good summary is provided in [5, Section 4.1]. This family forms a subset of ours, namely, $\psi_W(y) = \phi_{W,Wb}(y)$, i.e., with $v = \{W b\}$, and it is in fact known that, given any matrix weight $W$, that is the optimal choice for $v$ see, e.g., [6], where the result is shown for the equivalent mixed-integer rounding inequalities. However, this family depends on $b$, which would have complicated our discussion as well as the resulting algorithm, so we prefer to remain in this larger family to provide a more natural derivation.)

Our generalized GMI family satisfies the following.

Proposition 1. We have $\phi_{W,v} \in \mathcal{F}^{m', m}$ for any $W \in \mathbb{R}^{m \times m}$, $v \in [0, 1)^m$.

Proof. The function $g_v(y) = \min (y, \Delta \left( \frac{v}{1 - v} \right) [1 - y])$ is concave on $y \in [0, 1]^m$ and non-negative. Hence, by [7, Theorem 5.4], the function $g_v(y)$ is subadditive on $\mathbb{R}^m$. Then, $g_v([Wy_1 + y_2]) \leq g_v([Wy_1]) + g_v([Wy_2])$ for any $y_1, y_2$, so $h_{W,v} = g_v(y)$ is subadditive on $\mathbb{R}^m$. The UDDZ of $h_{W,v}$ is $\bar{h}_{W,v} = \max (Wy, -\Delta \left( \frac{v}{1 - v} \right) Wy)$, so that we can write
\[
\phi_{W,v}(y) = h_{W,v}(y) + \bar{h}_{W,v}(-I)y.
\]
Since $h_{W,v}$ is subadditive, and $y \mapsto \bar{h}_{W,v}(-I)y$ is linear, $\phi_{W,v}$ must be subadditive too.

Say $y_1 \leq y_2$. Then,
\[
\phi_{W,v}(y_1) = \phi_{W,v}(y_2 + (y_1 - y_2)) \leq \phi_{W,v}(y_2) + h_{W,v}(y_1 - y_2) + \bar{h}_{W,v}(-I)(y_1 - y_2) \leq \phi_{W,v}(y_2) + \bar{h}_{W,v}(-I)(y_2 - y_1) + \bar{h}_{W,v}(-I)(y_1 - y_2) = \phi_{W,v}(y_2)
\]
because $y_2 - y_1 \geq 0$. Hence, $\phi_{W,v}$ must be non-decreasing. Finally, $\phi_{W,v}(0) = 0$, so $\phi_{W,v} \in \mathcal{F}^{m', m'}$. □

In light of this proposition, functions $\phi_{W,v}(y)$ induce valid inequalities, and defining $f_{W,v}(y) =$
we had added apply the reasoning we just outlined to itself: if $\theta$ $K$

In fact, we can repeat this as many times we like, say $m$ rounds of GMI cuts corresponds to a specific se-

Moreover, this generalizes the classical GMI cuts, parameters vector $\theta_{GMI} = (W_1, v_1, \ldots, W_K, v_K)$ for problem (7), when $m_1 = m, \ldots, m_K = 2^{K-1}m$.

4. Continuous cutting plane optimization

One main reason why the weights $\theta_{GMI}$ of the classical GMI algorithm are desirable, besides their low computational cost, is that the lower bound obtained by solving the LP relaxation of problem (7) with dimensions $(m_1, \ldots, m_K) = (m, \ldots, 2^{K-1}m)$, that is

$$LP(f_\theta) = \min_{(x,z) \in \mathbb{R}^n} c'x + h'z \quad \text{s.t.} \quad f_\theta(A)x + \tilde{f}_\theta(G)z \geq f_\theta(b), \quad x, z \geq 0,$$

with $\theta = \theta_{GMI}$, tends to be high (see, e.g., [1]). A natural question is whether those weights are optimal: that is, does $\theta = \theta_{GMI}$ maximize the dual bound $LP(f_\theta)$ with $(m_1, \ldots, m_K) = (m, \ldots, 2^{K-1}m)$? We will see empirically this is not the case, even for $K = 1$.

Hence, we could instead consider the following.

Let us agree ahead of time that we will have $K$ rounds of generalized GMI valid inequalities, with $m_1, \ldots, m_K$ (not necessarily $m, \ldots, 2^{K-1}m$) inequalities in each round.

**Definition 2.** We call continuous cutting plane optimization the unconstrained nonlinear problem

$$\max_{\theta} LP(f_\theta).$$

That is, for a number of rounds $K$ and dimensions $m_1, \ldots, m_K$ fixed a priori, we try to find the weights that make $f_\theta$ yield the highest dual bound possible when used as a cut-generating function.

5. Deep subadditive primal optimization

The structure of the function $f_\theta$ we are optimizing in problem (8) is worth a closer look. This function has a very specific structure, which reminds one of deep neural networks in machine learning. Indeed, in the language of the field, we could describe the function $f_\theta$ as a neural network with $K$ layers $f_{W_\nu}(y) = ...$
in a greedy, layer-by-layer fashion. In this neural network, each layer $f_{W,k}$ corresponds to a round of valid inequalities, and have the specific form

$$f_{W,k} = \left[ y, \min \left(W_k y, \Delta \left(\frac{v_k}{1 - v_k}\right) [1 - (W_k y)] \right) \right] + \max \left(-W_k, \Delta \left(\frac{v_k}{1 - v_k}\right) W_k y \right).$$

for learnable parameters $W_k, v_k$. In contrast, our continuous cuts optimization problem (9) can be interpreted as training the same neural network, but in an end-to-end fashion (all layers at the same time). In other words, we maintain the same generalized GMI function (5) in our layers, but we continuously update the weights in the attempt to find better cuts.

Interestingly, this suggests that there is no strong reason why we should restrict ourselves to GMI cuts: the approach could very well be applied to any other continuously parametrized family of cuts that are induced by a subadditive function, such as Chvátal-Gomory cuts, or even others that do not correspond to any known family. We will discuss further this possibility in the conclusion.

6. Concrete algorithm

Our continuous cutting plane optimization problem (9), although unconstrained, is highly nonlinear and standard optimization approaches quickly run into difficulties. In particular, a challenge is that most $\theta$ parameters lead to valid inequalities that do not cut off the (optimal) vertices of the LP polytope, hence lead to zero gradients.

Hence, we propose instead a two-step heuristic to solve (9), detailed in Algorithm 1. The idea is to alternate between finding the optimal solution of the linear relaxation with the valid inequalities added, and changing the parameters by gradient descent so as to cut off this optimal solution. This algorithm seems to perform well in practice, and intuitively should lead to increase the dual bound LP($f_\emptyset$) as more and more LP solutions are cut off. On the other hand, it is also quite computationally expensive, although warm-starting the linear program from the previous optimal basis alleviates some of that cost.

In practice, we found it useful to add some noise to the point we aim to cut off: that is, instead of trying to cut the LP solution $x_{LP}^\star$, we try to cut a noisy
version \(x_{LP}^* + \beta \epsilon\), where \(\epsilon\) is a small amount of Gaussian noise and \(\beta\) is a hyperparameter (set to \(10^{-4}\) in all experiments). This helped in problems where degeneracy occurred in the linear relaxations.

The parametrization of our weights \(v\) require that they be contained in \([0, 1]\). This could be enforced, say, with projected gradient descent, but in practice we found it easier to enforce it to be in \((0, 1)\) by writing it as \(v = \text{sigmoid}(v') = 1/(1 + \exp(-v'))\) for some other parameter \(v' \in \mathbb{R}^m\), over which we optimize.

We propose two ways to initialize the weights of our inequalities.

1. We can warm-start the optimization by initializing the weights with the values obtained by running the classical GMI separation algorithm for \(K\) rounds. If \(m_k < 2^{k-1}m\), we can pick, for example, the weights corresponding to the \(m_k\) most violated cuts.

2. We can initialize from scratch with random weights. We attempted a few different possibilities, and our most successful approach was to initialize the rows of the \(W\) matrices with \(m_k\) mutually orthogonal random unit vectors, and the \(v' = \log(v) = \log(v/[1 - v])\) vectors from a \(N(0, I_{m_k})\) distribution.

Finally, the gradient step warrants further detail. Although in principle one could derive, through matrix calculus, closed form expressions for the gradient, we found it easier in practice to compute \(f_\theta(A), f_\theta(b)\) and their gradients using an automatic differentiation library (such as Pytorch).  

Now, running the classical GMI separation algorithm with an increasing number of rounds is also a way of increasing the dual bound over time. Therefore, the classical algorithm can really be seen in two ways: as a competitor for weight selection when \(K\) is fixed a priori, and as a more general competitor for obtaining dual bounds when \(K \rightarrow \infty\).

In the latter case, a few important differences emerge. Besides the fact we keep the number of cuts fixed (see (9)), another difference is that improvements are not monotone anymore. That is, there is no guarantee with our algorithm that the dual bound necessarily increases as we take more gradient steps, because when varying the GMI weights so as to cut the latest LP solution, previous points that were cut might not be cut anymore. In particular, it is possible that a previous LP solution gets “dislodged” and becomes feasible again for our generalized GMI inequalities after a gradient step, worsening the dual bound. Thus, our algorithm sacrifices monotonicity, although in practice this does not seem too high a price to pay for potentially improving the dual bound.

A more abstract way we can look at the situation is that in the classical GMI separation algorithm, the memory of the previous cuts is preserved in a trivial fashion by keeping the cuts in the LP, whereas in our algorithm, the memory of the previous cuts can be thought as preserved in the slowly evolving weights.

Note that the collection of cuts that one is forced to keep in a classical cutting plane approach depends on the sequence of fractional solutions that the algorithm needs to separate. However, we claim that such a sequence is somehow random, i.e., having very little guarantee of good convergence properties. In contrast, Algorithm [1] – driven by the quality of the bound and giving up on monotonicity – is able to distill a small cardinality set of cutting planes that are practically “robust” for a cloud of fractional solutions.

### 7. Experimental results

We now present two experiments that illustrate the performance of our approach. In each case, we im-
implemented our algorithm with the Pytorch 1.9.0 automatic differentiation framework [11], while we used Gurobi 9.1.1 [12] as LP solver, as well as for computing optimal MILP values for comparison. The experiments were run on Intel Xeon W-2145 3.70GHz CPU cores and a NVIDIA Titan RTX GPU.

7.1. First experiment

For our first experiment, we evaluate on 4 benchmarks: 100 randomly generated minimum set covering, combinatorial auctions, maximum independent set and capacitated facility location instances, that were generated using the Ecole python library [13] (following implementations from [14, 15, 16, 17], respectively.)

We report the results of optimizing a one-layer network with 32 inequalities, and a two-layer network with 32 inequalities each, illustrating the effect that deeper layers can have on performance. In addition, we show the results of starting from the classical GMI weights as well as from our random orthogonal initializer, as described in Section 6. Results over 10k gradient steps are reported in Figure 1. At any gradient step, we report the median optimality gap over all instances of the same family, where optimality gap (in our formulation) is defined as $(z^* - z)/|z^*|$ for a dual bound $z$ and the MILP optimal value $z^*$. We report the computational cost of the experiments in Appendix A.

As can be seen, in general our algorithm is able to learn and improve upon the classical GMI weights. This demonstrates that although Problem (9) is highly nonconvex, it is in fact possible to solve it to a degree where substantially better weights than the classical GMI weights can be found (albeit perhaps at great computational cost.) Moreover, using higher rank inequalities generally leads to better dual bounds, as expected.

We highlight however some less intuitive findings. First, the classical GMI weights seem to be fragile initial points, as the early gradient steps end up usually worsening the dual bound, until the algorithm can find a better position from which actual improvements eventually occur. This is particularly dramatic with the maximum independent set instances, where for the two-layer network it took on average 6k gradient steps until an improvement over the classical weights could be found. Incidentally, this illustrates rather dramatically our point in Section 6 that our approach does not guarantee monotone bound improvements.

Second, our random orthogonal initializer starts with the LP relaxation value, since the initial weights do not lead to inequalities that cut off any part of the LP polytope. However, interestingly, for at least two families (set covering and combinatorial auction instances), the performance eventually overtakes the curves initialized with the classical GMI weights. Thus, in at least some families, it appears that these initial inequalities might be better located for long-run optimization. At the other extreme, with capac-

---

Figure 1: Median optimality gap over the different instances as a function of the number of gradient steps taken by Algorithm 1. The yellow and orange lines represent the optimality gap with a random orthogonal initialization, with one and two layers, respectively. The blue and navy lines represent the optimality gap when warm-starting from the classical GMI weights, again with one and two layers respectively. The dashed lines represent the initial values in each setup, with the red dashed line being also the optimality gap of the LP relaxation.
iterated facility location instances the algorithm was never able to find effective cuts when starting from the randomly initialized weights. Thus, from this experiment we cannot conclude that one initialization method is always better than the other.

7.2. Second experiment

We now turn to our second experiment. In this scenario, we evaluated on 12 2-matching [18] instances derived from TSPLIB [19] and 18 small instances taken from MIPLIB3 [20]. These instances are more heterogeneous than in the first experiment, and the 2-matching problem are particularly interesting since rank-1 Chvátal-Gomory cuts are enough to compute their convex hull [21]. Since GMI inequalities dominate Chvátal-Gomory inequalities, this suggests that, in theory, a carefully optimized 1-layer network should be enough to solve the problems.

The instances in this experiment are harder than those in the first experiment, and we consequently use a larger setup, namely a two-layer network with 1024 inequalities in each layer. Again, we run the experiment with random orthogonal initialization, as well as initialization from the classical GMI weights. We report in Figure 2 the optimality gap achieved over each instance when running the algorithm for 10k and 50k gradient steps, on the 2-matching and MIPLIB3 instances, respectively. Appendix A summarizes the computational cost of this experiment.

As can be seen, when looking at individual instances the portrait is somewhat complex. When warm-starting with the classical GMI weights, the algorithm is often able to improve on the initial value, but the curves are not always smooth, and there might be periods of stalling. With random orthogonal initialization, the algorithm often did not manage to learn an improvement within the number of gradient steps provided, although for some it did. Thus, we can conclude that although the method appears in general feasible, it offers no guarantees.
8. Conclusions

In this paper, we proposed an alternative way of finding cutting planes for mixed-integer linear programs, where parameters are optimized so that a subadditive neural network acts as a good cut-generating function. Besides the theoretical interest of this point of view, and its intriguing potential connection with machine learning, we also show that the problem can be solved in practice, albeit at high computational cost.

As mentioned in Section 4, we chose in this paper to illustrate our ideas with the well-known Gomory mixed-integer cuts. In practice, however, we could really consider any family of cuts induced by subadditive, non-decreasing, centered functions, such as Chvátal-Gomory cuts, or some that might not have been previously considered in the literature. For example, we experimented with the family \( \zeta_{w^*}(y) = \log(1 + w^*(y)) \). We found that on the 2-matching and small MIPLIB3 instances, these “logarithmic” GMI cuts often yielded superior results, which we believe could be due to yielding a loss landscape that is easier to navigate for gradient descent. Thus, there is a wide room for experimentation, and it is possible that the best families for a “continuous” approach are quite different than those that were best for classical approaches based on sequential LP solving.

Moreover, although this work does not represent machine learning per se – with machine learning being defined as the study of statistical generalization – it is clear that the connection with deep neural networks suggests many potential ideas that could be applied to this line of work. To name a simple one, one could imagine adding a \( L_1 \) regularizer term in problem (9) as to encourage the weights, hence the cuts, to be sparse. We do not doubt that many other interesting ideas could be found.

Acknowledgments

The authors are indebted to Oktay Günlük and many members of the Canada Excellence Research Chair in “Data Science for Real-time Decision-making” at Polytechnique Montréal where most of this work has been conducted while the second author was the chair holder. We are indebted to an anonymous referee for their very careful reading and useful suggestions. We are also grateful to the feedback and discussion during the IPAM “Artificial Intelligence and Discrete Optimization” Workshop (February 27 - March 3, 2023).

References

[1] A. Lodi, Mixed integer programming computation, in: 50 years of integer programming 1958-2008, Springer, Berlin, Heidelberg, 2010, pp. 619–645.
[2] M. Guzelsoy, T. K. Ralphs, Duality for mixed-integer linear programs, International Journal of Operations Research 4 (3) (2007) 118–137.
[3] R. Gomory, An algorithm for the mixed integer problem, Tech. Rep. RM-2597, The Rand Corporation (1960).
[4] M. Fischetti, C. Saturni, Mixed-integer cuts from cyclic groups, Mathematical Programming 109 (1) (2007) 27–53.
[5] G. Cornuéjols, Valid inequalities for mixed integer linear programs, Mathematical Programming 112 (1) (2008) 3–44.
[6] S. Dash, O. Günlük, A. Lodi, MIR closures of polyhedral sets, Mathematical Programming 121 (1) (2010) 33–60.
[7] J. Matkowski, Subadditive periodic functions, Opuscula Mathematica 31 (1) (2011) 75–96.
[8] R. Jeroslow, Cutting-plane theory: Algebraic methods, Discrete Mathematics 23 (2) (1978) 121–150.
[9] R. G. Jeroslow, Minimal inequalities, Mathematical Programming 17 (1) (1979) 1–15.
[10] L. A. Wolsey, Integer programming duality: Price functions and sensitivity analysis, Mathematical Programming 20 (1) (1981) 173–195.
[11] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., Pytorch: An imperative style, high-performance deep learning library, Advances in Neural Information Processing Systems 32 (2019) 8026–8037.
[12] Gurobi Optimization LLC, Gurobi Optimizer Reference Manual (2020).
[13] A. Prouvost, J. Dumouchelle, L. Scavuzzo, M. Gasse, D. Chételat, A. Lodi, Ecole: A Gym-like library for machine learning in combinatorial optimization solvers, in: Learning Meets Combinatorial Algorithms Workshop at NeurIPS 2020, 2020.
[14] E. Balas, A. Ho, Set covering algorithms using cutting planes, heuristics, and subgradient optimization: a computational study, in: Combinatorial Optimization, Springer, 1980, pp. 37–60.
[15] K. Leyton-Brown, M. Pearson, Y. Shoham, Towards a universal test suite for combinatorial auction algorithms, in: Proceedings of the 2nd ACM Conference on Electronic Commerce, 2000, pp. 66–76.
[16] D. Bergman, A. A. Cire, W.-J. Van Hoeve, J. Hooker, Decision diagrams for optimization, Vol. 1, Springer, 2016.
[17] G. Cornuéjols, R. Sridharan, J.-M. Thizy, A comparison of heuristics and relaxations for the capacitated plant loca-
| Set Cover | Cautions | Indset | Facilities | 2-Matching | Small-Miplib3 |
|----------|----------|--------|------------|------------|--------------|
| Randomly initialized, 1 layer | 29.54 ± 190.11 | 22.14 ± 146.06 | 21.71 ± 143.24 | 18.39 ± 102.77 | 49.87 ± 27.01 | 354.10 ± 370.82 |
| Randomly initialized, 2 layers | 34.30 ± 190.16 | 26.10 ± 146.78 | 24.29 ± 142.83 | 16.92 ± 87.38 | 133.06 ± 245.66 | 534.88 ± 440.83 |
| Warm-started, 1 layer | 20.36 ± 50.89 | 29.60 ± 164.28 | 11.87 ± 30.18 | 13.67 ± 75.99 | 145.97 ± 240.40 | 431.59 ± 338.40 |
| Warm-started, 2 layers | 25.64 ± 70.78 | 13.13 ± 13.63 | 20.90 ± 61.08 | 5.14 ± 8.23 | 115.87 ± 136.40 | 519.75 ± 357.89 |
| Gurobi | 0.0040 ± 0.0044 | 0.0089 ± 0.0060 | 0.0008 ± 0.0007 | 0.0002 ± 0.0001 | 0.0001 ± 0.0001 | 0.1260 ± 0.5541 |

Table 1: Average time in minutes (± standard deviation) required to achieve the desired number of gradient steps in Experiments 7.1 and 7.2. We also include the average time taken for Gurobi to solve the problems to optimality.

Appendix A. Computational cost

For reference, we report in Table A the computational cost of the experiments, as well as the times that Gurobi takes to solve these problems to optimality. As can be seen, although our method is able to find good dual bounds, the computational cost is prohibitive compared to simply feeding the instances to a standard combinatorial optimization solver. Admittedly, we have not explored any path for reducing such computational effort, and we clearly consider this an important future step.