SIAM J. OPTIM. © 2020 Society for Industrial and Applied Mathematics
Vol. 30, No. 1, pp. 1007–1032

PROJECTIVE CUTTING-PLANES[*]

DANIEL PORUMBEL[†]

Abstract. Given a polytope $\mathscr{P}$, an interior point $\mathbf{x} \in \mathscr{P}$, and a direction $\mathbf{d} \in \mathbb{R}^n$, the projection of $\mathbf{x}$ along $\mathbf{d}$ asks to find the maximum step length t^* such that $\mathbf{x} + t^*\mathbf{d} \in \mathscr{P}$; we say $\mathbf{x} + t^*\mathbf{d}$ is the pierce point obtained by projection. In [D. Porumbel, *Math. Program.*, 155 (2016), pp. 147–197], we solely explored the idea of projecting the origin $\mathbf{0}_n$ along integer directions only, focusing on dual polytopes $\mathscr{P}$ in `Column Generation` models. This work addresses a more general projection subproblem, considering arbitrary interior points $\mathbf{x} \in \mathscr{P}$ and arbitrary noninteger directions $\mathbf{d} \in \mathbb{R}^n$, in areas beyond `Column Generation`. The projection subproblem generalizes the separation subproblem of the well-known `Cutting-Planes`. We propose a new algorithm, `Projective Cutting-Planes`, that relies on this projection subproblem to optimize over polytopes $\mathscr{P}$ with prohibitively many constraints. At each iteration, this new algorithm selects a point $\mathbf{x}_{\text{new}}$ on the segment joining the points $\mathbf{x}$ and $\mathbf{x}+t^*\mathbf{d}$ determined at the previous iteration. Then, it projects $\mathbf{x}_{\text{new}}$ along the direction $\mathbf{d}_{\text{new}}$ pointing towards the current optimal (outer) solution (of the current outer approximation of $\mathscr{P}$), so as to generate a new pierce point $\mathbf{x}_{\text{new}} + t^*_{\text{new}}\mathbf{d}_{\text{new}}$ and a new constraint of $\mathscr{P}$. By reoptimizing the linear program enriched with this new constraint, the algorithm finds a new current optimal (outer) solution and moves to the next iteration by updating $\mathbf{x} = \mathbf{x}_{\text{new}}$ and $\mathbf{d} = \mathbf{d}_{\text{new}}$. Compared to `Cutting-Planes`, the main advantage of `Projective Cutting-Planes` is that it has a *built-in* functionality to generate a feasible inner solution new $+\, t^*\mathbf{d}$ at each iteration. These inner solutions converge iteratively to an optimal solution $\texttt{opt}(\mathscr{P})$, and so `Projective Cutting-Planes` is more similar to an interior point method than to the Simplex method. Numerical experiments in different optimization settings confirm the potential of the proposed ideas.

Key words. `Cutting-Planes`, separation subproblem, polytopes, `Column Generation`, robust optimization

AMS subject classifications. 65K05, 97N60, 90C05

DOI. 10.1137/19M1272652

1. Introduction.

Optimizing linear programs (LPs) with prohibitively many constraints has a long history in mathematical optimization. The `Cutting-Planes` algorithm maintains at each iteration `it` an outer approximation $\mathscr{P}_{\texttt{it}}$ of $\mathscr{P}$, i.e., a polytope $\mathscr{P}_{\texttt{it}}$ defined only by a subset of the constraints of $\mathscr{P}$, so that $\mathscr{P}_{\texttt{it}} \supseteq \mathscr{P}$. The most canonical `Cutting-Planes` can be seen as an outer method in the sense that it converges towards an optimal solution $\texttt{opt}(\mathscr{P})$ through a sequence of outer (infeasible) solutions, with no built-in functionality to generate inner solutions. In contrast, an inner method constructs a sequence of inner feasible solutions $\mathbf{x}_{\texttt{it}}$ that converge towards $\texttt{opt}(\mathscr{P})$ along the iterations `it`. The proposed `Projective Cutting-Planes` is both an inner and an outer method in the sense that it generates a convergent sequence of both inner and outer solutions. We refer the reader to (section 1.1 of) [13] for more information and comparisons of inner methods and outer methods.

The proposed algorithm relies on an iterative operation of projecting an interior point onto facets of $\mathscr{P}$, as illustrated in Figure 1. At each iteration `it`, an inner solution $\mathbf{x}_{\texttt{it}} \in \mathscr{P}$ is projected towards the direction $\mathbf{d}_{\texttt{it}}$ of the current optimal outer solution $\texttt{opt}(\mathscr{P}_{\texttt{it}-1})$, i.e., we take $\mathbf{d}_{\texttt{it}} = \texttt{opt}(\mathscr{P}_{\texttt{it}-1}) - \mathbf{x}_{\texttt{it}}$. The projection subproblem asks to determine $t^*_{\texttt{it}} = \max\{t : \mathbf{x}_{\texttt{it}} + t\mathbf{d}_{\texttt{it}} \in \mathscr{P}\}$. For this, one has to find

[*]Received by the editors July 5, 2019; accepted for publication (in revised form) December 23, 2019; published electronically March 26, 2020.
https://doi.org/10.1137/19M1272652

[†]CEDRIC CS Lab, CNAM, Paris, 75003, France (daniel.porumbel@cnam.fr).

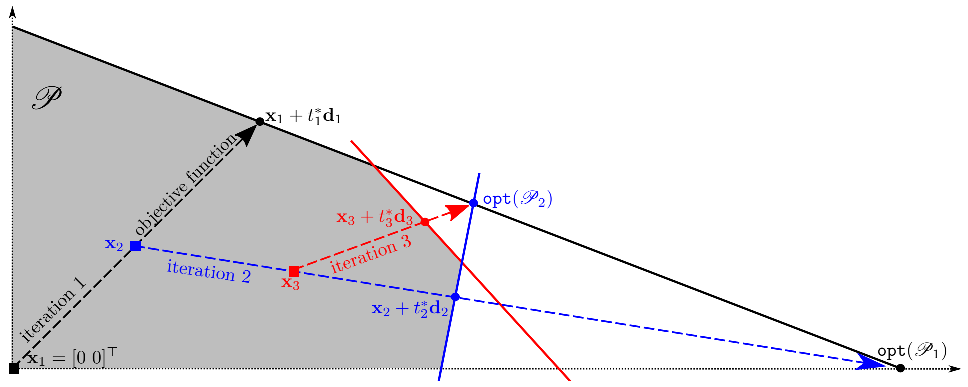
$\mathscr{P}$
$\mathbf{x}_1 + t_1^*\mathbf{d}_1$
objective function
iteration 1
$\mathbf{x}_1 = [0\ 0]^\top$
$\mathbf{x}_2$
iteration 2
$\mathbf{x}_3$
iteration 3
$\mathbf{x}_3 + t_3^*\mathbf{d}_3$
$\mathtt{opt}(\mathscr{P}_2)$
$\mathbf{x}_2 + t_2^*\mathbf{d}_2$
$\mathtt{opt}(\mathscr{P}_1)$

FIG. 1. *The first three iterations of the* `Projective Cutting-Planes` *on an LP with two variables. At the first iteration, the projection subproblem projects* $\mathbf{x}_1 = \mathbf{0} = [0\ 0]^\top$ *along the objective function, as depicted by the black dashed arrow. At iteration* $\mathtt{it} = 2$*, the midpoint* $\mathbf{x}_2$ *of this black arrow is projected towards the optimal outer solution* $\mathtt{opt}(\mathscr{P}_1)$*; at iteration* 1*, the outer approximation* $\mathscr{P}_1 \supset \mathscr{P}$ *only contains the largest triangle. This generates a second facet (blue solid line) that is added to the facets of* $\mathscr{P}_1$ *to construct* $\mathscr{P}_2$*. The third projection is represented by the short arrow in red. (Color available online.)*

the pierce (hit) point $\mathbf{x}_{\mathtt{it}} + t^*_{\mathtt{it}}\mathbf{d}_{\mathtt{it}}$ and a (first-hit) constraint of $\mathscr{P}$, which is added to the constraints of $\mathscr{P}_{\mathtt{it}-1}$ to construct $\mathscr{P}_{\mathtt{it}}$. At the next iteration $\mathtt{it}+1$, the proposed `Projective Cutting-Planes` takes a new interior point $\mathbf{x}_{\mathtt{it}+1}$ on the segment joining $\mathbf{x}_{\mathtt{it}}$ and $\mathbf{x}_{\mathtt{it}} + t^*_{\mathtt{it}}\mathbf{d}_{\mathtt{it}}$, and it projects $\mathbf{x}_{\mathtt{it}+1}$ along $\mathbf{d}_{\mathtt{it}+1} = \mathtt{opt}(\mathscr{P}_{\mathtt{it}}) - \mathbf{x}_{\mathtt{it}+1}$.

To determine $t^* = \max\{t : \mathbf{x} + t\mathbf{d} \in \mathscr{P}\}$, one has to find a (first-hit) constraint satisfied with equality by $\mathbf{x} + t^*\mathbf{d}$. This projection subproblem implicitly solves the separation subproblem for all points $\mathbf{x} + t\mathbf{d}$ with $t \in \mathbb{R}_+$, because the above first-hit constraint separates all solutions $\mathbf{x}+t\mathbf{d}$ with $t > t^*$ and proves $\mathbf{x}+t\mathbf{d} \in \mathscr{P}\ \forall t \in [0, t^*]$. A simplified version of the projection subproblem limited to $\mathbf{x} = \mathbf{0}_n$ was already studied in our previous papers on `Column Generation` [13] and Benders decomposition models [14]. The current work seeks maximum generality in terms of projections: we will project arbitrary interior points $\mathbf{x} \in \mathscr{P}$ along arbitrary directions $\mathbf{d} \in \mathbb{R}^n$, addressing more diverse problems than [13] and [14] together.

The proposed algorithm is reminiscent of an interior point method (IPM) in the sense that it generates a sequence of interior points that converge to the optimal solution. An IPM moves from solution to solution by advancing along a Newton direction at each iteration, in an attempt to solve first order optimality conditions [7]. Advancing along a Newton direction is not really equivalent to performing a projection, because a projection executes a full step-length (advancing up to the pierce point) while a Newton step in an IPM does not even advance to fully solve the first order conditions—since these conditions correspond to a primal objective function penalized by a barrier term that only vanishes at the last iteration. Certain IPMs for (dual) LPs with prohibitively many constraints (in `Column Generation`) generate well-centered dual solutions along the iterations by keeping them in the proximity of a central path [8, section 3.3]. This shares certain goals with the construction of the feasible solutions $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3, \dots$ in `Projective Cutting-Planes`. However, each solution of the above central path belongs to some $\mathscr{P}_{\mathtt{it}} \supset \mathscr{P}$, but not necessarily to $\mathscr{P}$.

Since it generalizes the separation subproblem, the projection subproblem may seem computationally (far) more expensive, but we will see that this is not always the case. We will present in section 2.2 an overview of several techniques that can bring us very close to designing a projection algorithm as fast as the separation one. A first technique simply consists of generalizing the separation algorithm when this does not induce a significant loss in complexity. We will exemplify this idea on a robust optimization problem where both subproblems have the same computational bottleneck (scanning all nominal constraints). Another technique applies to the numerous problems in which the constraints of $\mathscr{P}$ are associated to the feasible solutions of an auxiliary LP or of an integer LP (ILP), as often happens in Benders reformulation models or, respectively, in `Column Generation`. In this case, the projection subproblem can be written as a linear-fractional program: minimize a ratio of two linear functions subject to linear constraints. Such (continuous, resp., discrete) programs can be cast into classical (continuous, resp., discrete) LPs using the Charnes–Cooper transformation [3], and so, they become as difficult as the separation subproblem.

We will show that a standard `Projective Cutting-Planes` implementation can quite easily outperform a standard `Cutting-Planes` and even certain state-of-the-art enhanced `Cutting-Planes`. For instance, for the (well-studied) graph coloring problem, `Projective Cutting-Planes` is able to reach lower bounds that have never been reported before (Remark 6, page 1028). For the Benders decomposition, we will also compare to the performances of an enhanced `Cutting-Planes` which seem dominated by `Projective Cutting-Planes`. Further experiments on *(Multiple-Length) Cutting-Stock* from the follow-up paper [15, section 3.2] confirm that `Projective Cutting-Planes` can yield an acceleration factor significantly greater than the one that can be obtained by traditional stabilization techniques (which is generally below 20%).

However, the (only) goal of `Projective Cutting-Planes` is *not* to compete with `Cutting-Planes`, and this work was not meant to be a competition paper. It is more important that the new algorithm has certain features that do not exist in `Cutting-Planes`; e.g., it generates feasible solutions along the iterations, it can eliminate the "yo-yo" effects arising very often (if not always) in `Column Generation` (see Figure 3), and degeneracy risks are drastically reduced [15, Remark 2].

The remainder of the paper is organized as follows. Section 2 provides a detailed description of the generic `Projective Cutting-Planes`. Section 3 illustrates the application of this algorithm on different problems in which $\mathscr{P}$ is either a primal (master) polytope or a dual polytope (in `Column Generation`). Section 4 is devoted to numerical results, followed by conclusions in section 5.

2. Algorithmic description of Projective Cutting-Planes.

Given a set of (unmanageably many) constraints $\mathcal{A}$, this paper is focused on solving[1]

$$\max\left\{\mathbf{b}^\top\mathbf{x} : \mathbf{a}^\top\mathbf{x} \le c_a \ \forall(\mathbf{a}, c_a) \in \mathcal{A}\right\} = \max\left\{\mathbf{b}^\top\mathbf{x} : \mathbf{x} \in \mathscr{P}\right\}. \tag{2.1}$$

The standard `Cutting-Planes` for solving this LP maintains at each iteration `it` an outer approximation $\mathscr{P}_{\mathtt{it}} \supset \mathscr{P}$ of $\mathscr{P}$ obtained by restricting the constraint set $\mathcal{A}$ to a subset $\mathcal{A}_{\mathtt{it}}$. To (try to) separate the current optimal solution $\mathbf{x}^{\mathrm{out}} = \mathtt{opt}(\mathscr{P}_{\mathtt{it}})$

[1]In fact, we will also address a few variations of (2.1). As such, the Benders reformulation model (3.2a)–(3.2c) uses integer variables $\mathbf{x} \in \mathbb{Z}^n_+$. When (2.1) is a dual LP obtained after relaxing an integer `Column Generation` LP, the goal is actually to find the best *rounded-up* objective value, i.e., "$\max \mathbf{b}^\top\mathbf{x}$" can be replaced by "$\max \lceil \mathbf{b}^\top\mathbf{x} \rceil$." One can also use "$\min \mathbf{b}^\top\mathbf{x}$" instead of "$\max \mathbf{b}^\top\mathbf{x}$" with no impact on the algorithm design. One can also envisage addressing the case of infinite sets $\mathcal{A}$.

of $\mathscr{P}_{\mathtt{it}}$, the most standard `Cutting-Planes` usually solves the separation subproblem $\min_{(\mathbf{a},c_a)\in\mathcal{A}} c_a - \mathbf{a}^\top \mathbf{x}^{\text{out}}$. If the optimum value of this subproblem is less than 0 for some $(\overline{\mathbf{a}}, \overline{c_a}) \in \mathcal{A}$, then $\mathbf{x}^{\text{out}}$ is infeasible. In this case, the `Cutting-Planes` method inserts $\overline{\mathbf{a}}^\top \mathbf{x} \leq \overline{c_a}$ into the current constraint set (i.e., it performs $\mathcal{A}_{\mathtt{it}+1} = \mathcal{A}_{\mathtt{it}} \cup \{(\overline{\mathbf{a}}, \overline{c_a})\}$) so as to construct a new, more refined, outer approximation $\mathscr{P}_{\mathtt{it}+1}$ and to separate $\mathbf{x}^{\text{out}} \notin \mathscr{P}_{\mathtt{it}+1}$. The process is repeated by (re)optimizing over $\mathscr{P}_{\mathtt{it}+1}$ at the next iteration, until the current optimal outer solution $\mathbf{x}^{\text{out}}$ becomes optimal (nonseparable).

We propose replacing the above separation subproblem with the following one.

Definition 2.1. *(Projection subproblem) Given an interior point $\mathbf{x} \in \mathscr{P}$ and a direction $\mathbf{d} \in \mathbb{R}^n$, the projection subproblem* project$(\mathbf{x} \to \mathbf{d})$ *asks to find the following:*

(1) *The maximum step length t^* such that $\mathbf{x} + t^*\mathbf{d}$ is feasible inside $\mathscr{P}$, i.e., $t^* = \max\{t \geq 0 : \mathbf{x} + t\mathbf{d} \in \mathscr{P}\}$. The solution $\mathbf{x} + t^*\mathbf{d}$ is referred to as the pierce point. If $\mathbf{x} + t\mathbf{d}$ is a ray of $\mathscr{P}$, the subproblem returns $t^* = \infty$.*

(2) *A first-hit constraint $(\mathbf{a}, c_a) \in \mathcal{A}$ satisfied with equality by the pierce point, i.e., such that $\mathbf{a}^\top(\mathbf{x} + t^*\mathbf{d}) = c_a$; such a constraint certainly exists if $t^* \neq \infty$.*

At the very first iteration, `Projective Cutting-Planes` can start by performing a projection along $\mathbf{d}_1 = \mathbf{b}$ so as to directly advance along the direction with the fastest rate of objective function improvement, or use any problem-specific direction $\mathbf{d}_1$. An initial feasible (inner) solution $\mathbf{x}_1$ is always needed because we do not focus on problems for which it is difficult to decide whether (2.1) is feasible or not.[2]

By solving `project`$(\mathbf{x}_1 \to \mathbf{d}_1)$ at iteration $\mathtt{it} = 1$, `Projective Cutting-Planes` determines the first pierce point $\mathbf{x}_1 + t_1^*\mathbf{d}_1$ and generates a first-hit constraint $(\mathbf{a}, c_a) \in \mathcal{A}$. After updating $\mathcal{A}_1 = \mathcal{A}_0 \cup \{(\mathbf{a}, c_a)\}$, the first outer approximation $\mathscr{P}_1$ is constructed. Note that $\mathcal{A}_0$ may contain some simple initial constraints like $\mathbf{x} \geq \mathbf{0}_n$. The proposed method then executes the following steps at each iteration $\mathtt{it} \geq 2$:

1. Select an inner solution $\mathbf{x}_{\mathtt{it}}$ on the segment joining $\mathbf{x}_{\mathtt{it}-1}$ and $\mathbf{x}_{\mathtt{it}-1} + t^*_{\mathtt{it}-1}\mathbf{d}_{\mathtt{it}-1}$, i.e., on the segment between the previous inner solution and the last pierce point.
2. Take the direction $\mathbf{d}_{\mathtt{it}} = \mathtt{opt}(\mathscr{P}_{\mathtt{it}-1}) - \mathbf{x}_{\mathtt{it}}$ pointing towards the current optimal (outer) solution $\mathtt{opt}(\mathscr{P}_{\mathtt{it}-1})$.[3] Given that $\mathscr{P}_{\mathtt{it}-1} \supseteq \mathscr{P} \ni \mathbf{x}_{\mathtt{it}}$, we obtain that if $\mathbf{x}_{\mathtt{it}}$ is strictly interior, then the objective value can *only strictly improve* by advancing along $\mathbf{x}_{\mathtt{it}} \to \mathbf{d}_{\mathtt{it}}$; under these conditions, it is impossible to execute degenerate iterations that keep the objective value constant as in standard `Cutting-Planes`.
3. Solve the projection subproblem `project`$(\mathbf{x}_{\mathtt{it}} \to \mathbf{d}_{\mathtt{it}})$ to determine the maximum step length $t^*_{\mathtt{it}}$, the pierce point $\mathbf{x}_{\mathtt{it}} + t^*_{\mathtt{it}}\mathbf{d}_{\mathtt{it}}$, and a first-hit constraint $(\overline{\mathbf{a}}, \overline{c_a}) \in \mathcal{A}$.
4. If $t^*_{\mathtt{it}} \geq 1$, return $\mathtt{opt}(\mathscr{P}_{\mathtt{it}-1})$ as an optimal solution of (2.1) over $\mathscr{P}$.
   If $t^*_{\mathtt{it}} < 1$, then current optimal solution $\mathtt{opt}(\mathscr{P}_{\mathtt{it}-1})$ can be separated, and so, `Projective Cutting-Planes` performs the following:
   – set $\mathcal{A}_{\mathtt{it}} = \mathcal{A}_{\mathtt{it}-1} \cup \{(\overline{\mathbf{a}}, \overline{c_a})\}$ to obtain a new enlarged constraint set, corresponding to a more refined outer approximation $\mathscr{P}_{\mathtt{it}}$ that excludes $\mathtt{opt}(\mathscr{P}_{\mathtt{it}-1})$.
   – calculate a new current optimal outer solution $\mathtt{opt}(\mathscr{P}_{\mathtt{it}})$ by (re)optimizing over $\mathscr{P}_{\mathtt{it}}$.
   – if $\mathbf{x}_{\mathtt{it}} + t^*_{\mathtt{it}}\mathbf{d}_{\mathtt{it}}$ and $\mathtt{opt}(\mathscr{P}_{\mathtt{it}})$ are close enough (in terms of objective value), stop and return $\mathtt{opt}(\mathscr{P}_{\mathtt{it}})$. For instance, if (2.1) is a relaxation of an integer program

[2]For instance, in most `Column Generation` models, $\mathbf{x}_1 = \mathbf{0}_n$ is feasible. If $\mathbf{0}_n$ is infeasible, one may determine the initial $\mathbf{x}_1$ using any problem-specific heuristic. If it is particularly difficult to find a feasible solution, then the underlying problem is really outside the scope of this work.

[3]As for the standard `Cutting-Planes`, $\mathtt{opt}(\mathscr{P}_{\mathtt{it}-1})$ could be an extreme ray $\mathbf{r}$ of $\mathscr{P}_{\mathtt{it}-1}$ such that $\{\mathbf{x} + \lambda\mathbf{r} : \lambda \geq 0\} \subset \mathscr{P}_{\mathtt{it}-1}\ \forall \mathbf{x} \in \mathscr{P}_{\mathtt{it}}$; considering $\mathtt{opt}(\mathscr{P}_{\mathtt{it}-1}) = \{\mathbf{x}_{\mathtt{it}} + \lambda\mathbf{r} : \lambda \geq 0\} \subset \mathscr{P}_{\mathtt{it}-1}$, we obtain $\mathbf{d}_{\mathtt{it}} = \mathbf{r}$. If `project`$(\mathbf{x}_{\mathtt{it}} \to \mathbf{r})$ returns ∞, the algorithm returns that (2.1) is unbounded.

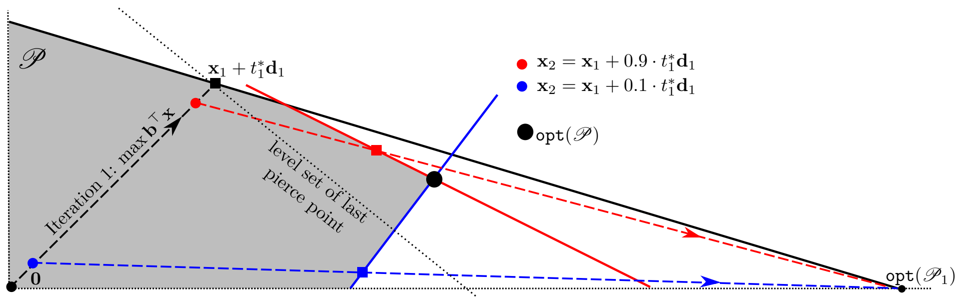
$\mathscr{P}$
$\mathbf{x}_1 + t_1^*\mathbf{d}_1$
Iteration 1: $\max \mathbf{b}^\top \mathbf{x}$
level set of last pierce point
$\mathbf{0}$
● $\mathbf{x}_2 = \mathbf{x}_1 + 0.9 \cdot t_1^*\mathbf{d}_1$
● $\mathbf{x}_2 = \mathbf{x}_1 + 0.1 \cdot t_1^*\mathbf{d}_1$
● $\mathtt{opt}(\mathscr{P})$
$\mathtt{opt}(\mathscr{P}_1)$

FIG. 2. *Intuitive illustration of two different choices of the interior point $\mathbf{x}_2$ at iteration 2. The red choice is more aggressive, while the blue one is more cautious. (Color available online.)*

(as in `Column Generation`), the stopping condition is to reach the same rounded-up value of the lower and the upper bounds.

– repeat from Step 1 after updating $\mathtt{it} \leftarrow \mathtt{it} + 1$.

The above algorithm is finitely convergent because it implicitly solves a separation subproblem on $\mathtt{opt}(\mathscr{P}_{\mathtt{it}-1})$ at each iteration $\mathtt{it}$, generalizing the standard `Cutting-Planes`. As hinted at in Step 4, if the projection subproblem returns $t^*_{\mathtt{it}} < 1$, the solution $\mathtt{opt}(\mathscr{P}_{\mathtt{it}-1})$ is certainly separated by the first-hit constraint $(\overline{\mathbf{a}}, \overline{c_a})$. In pure theory, in the worst case, the proposed method ends up enumerating all constraints of $\mathscr{P}$, and it then eventually returns $\mathtt{opt}(\mathscr{P})$. The fact that this convergence proof is very short is not completely fortuitous. Building on previous work [13, 14] with longer (convergence) theorems, the new `Projective Cutting-Planes` has been deliberately designed to simplify all proofs as much as possible.

2.1. Choosing the interior point $\mathbf{x}_{\mathtt{it}}$ at each iteration $\mathtt{it}$.

Just like the standard `Cutting-Planes`, `Projective Cutting-Planes` is a rather generic algorithm that allows a number of problem-specific adaptations.

A key question is the choice of the interior point $\mathbf{x}_{\mathtt{it}}$ at (Step 1 of) each iteration $\mathtt{it}$. One might attempt to choose the best feasible solution found up to iteration $\mathtt{it}$ (the last pierce point) using the formula $\mathbf{x}_{\mathtt{it}} = \mathbf{x}_{\mathtt{it}-1} + t^*_{\mathtt{it}-1}\mathbf{d}_{\mathtt{it}-1}$. While this aggressive strategy may perform well in certain settings, it may also lead to poor results in the long run for many problems, partly because $\mathbf{x}_{\mathtt{it}}$ can fluctuate too much from iteration to iteration (this is referred to as the bang-bang effect and will be further studied in [15, section 3.3]). In practice, the best results have often been obtained by choosing $\mathbf{x}_{\mathtt{it}} = \mathbf{x}_{\mathtt{it}-1} + \alpha t^*_{\mathtt{it}-1}\mathbf{d}_{\mathtt{it}-1}$ with $\alpha < 1$; a value of $\alpha = 1$ does not seem very effective for any problem studied in this work (or in [15]) except graph coloring. This is reminiscent of interior point methods for linear programming that usually avoid touching the boundary of the polytope before fully converging [7].

However, choosing $\mathbf{x}_{\mathtt{it}} = \mathbf{x}_{\mathtt{it}-1} + t^*_{\mathtt{it}-1}\mathbf{d}_{\mathtt{it}-1}$ when possible (e.g., for graph coloring) has the advantage that it enables the resulting `Projective Cutting-Planes` to improve the objective value $\mathbf{b}^\top\mathbf{x}_{\mathtt{it}}$ at each new iteration $\mathtt{it}$—because in such cases each projection $\mathbf{x}_{\mathtt{it}} \to \mathbf{d}_{\mathtt{it}}$ can only increase the objective value, as indicated in Step 2 above. As such, the lower bounds of this aggressive `Projective Cutting-Planes` variant are monotonically increasing (Figure 3), i.e., they do not exhibit any "yo-yo" effect with ups and downs like in most (if not all) `Column Generation` algorithms.

Figure 2 illustrates the difference between an aggressive choice (large α) and a "cautious" or well-centered choice (small α). The red circle represents an aggressive definition of $\mathbf{x}_2$ associated to a large α, so that $\mathbf{x}_2$ is very close to the last pierce point

$\mathbf{x}_1 + t_1^*\mathbf{d}_1$. Such a choice enables the projection subproblem at iteration 2 to easily exceed the objective value of the last pierce point $\mathbf{x}_1 + t_1^*\mathbf{d}_1$ by only advancing a little from $\mathbf{x}_2$ towards $\texttt{opt}(\mathscr{P}_1)$; see how rapidly the red dashed arrow crosses the black dotted line, i.e., the level set of the last pierce point $\{\mathbf{x} \in \mathbb{R}^2_+ : \mathbf{b}^\top\mathbf{x} = \mathbf{b}^\top(\mathbf{x}_1 + t_1^*\mathbf{d}_1)\}$. The blue circle represents a choice of a point $\mathbf{x}_2$ closer to $\mathbf{0}_n$: it is more difficult to reach the level set of $\mathbf{x}_1 + t_1^*\mathbf{d}_1$ by advancing from this $\mathbf{x}_2$ towards $\texttt{opt}(\mathscr{P}_1)$, but this blue projection can lead to a stronger (blue) constraint, i.e., the blue solid line cuts off a larger area of $\mathscr{P}_1$ (i.e., of the largest triangle) than the red solid line.

2.2. Techniques for designing a fast projection algorithm. A challenging aspect when implementing `Projective Cutting-Planes` is the design of a fast projection algorithm, because the iterations of a successful `Projective Cutting-Planes` should not be significantly slower than the iterations of the standard `Cutting-Planes`. For instance, if the projection iterations were two to three times slower than the separation iterations, the `Projective Cutting-Planes` could be too slow, i.e., it could remain slower than the standard `Cutting-Planes` even if it converged in half the number of iterations. Although the projection subproblem generalizes the separation one, we present below several techniques that lead to designing a projection algorithm that competes (very) tightly with the separation algorithm in terms of computational speed.

Let us first describe how the projection subproblem $\texttt{project}(\mathbf{x} \to \mathbf{d})$ reduces to minimizing the following fractional program (for any $\mathbf{x} \in \mathscr{P}$ and any $\mathbf{d} \in \mathbb{R}^n$):

$$t^* = \min\left\{\frac{c_a - \mathbf{a}^\top\mathbf{x}}{\mathbf{a}^\top\mathbf{d}} : (\mathbf{a}, c_a) \in \mathcal{A},\ \mathbf{d}^\top\mathbf{a} > 0\right\}. \tag{2.2}$$

We have to show $\mathbf{x}+t\mathbf{d} \in \mathscr{P}\ \forall t \in [0, t^*]$, i.e., $\mathbf{a}^\top(\mathbf{x}+t\mathbf{d}) \le c_a\ \forall(\mathbf{a}, c_a) \in \mathcal{A}\ \forall t \in [0, t^*]$. If $\mathbf{a}^\top\mathbf{d} \le 0$, then $\mathbf{a}^\top(\mathbf{x} + t\mathbf{d}) \le \mathbf{a}^\top\mathbf{x} \le c_a$ actually holds for all $t \in [0, \infty]$, because $\mathbf{x} \in \mathscr{P} \implies \mathbf{a}^\top\mathbf{x} \le c_a$. Otherwise, if $\mathbf{a}^\top\mathbf{d} > 0$, then $\mathbf{a}^\top(\mathbf{x} + t\mathbf{d}) \le c_a$ is equivalent to $t \le \frac{c_a - \mathbf{a}^\top\mathbf{x}}{\mathbf{a}^\top\mathbf{d}}$, which is true for any $t \le t^*$, because t^* minimizes the above ratio in (2.2). As such, we will only focus on $(\mathbf{a}, c_a) \in \mathcal{A}$ such that $\mathbf{a}^\top\mathbf{d} > 0$ when designing the projection algorithm. Finally, $\mathbf{x} + t^*\mathbf{d}$ belongs to the boundary of $\mathscr{P}$ because it satisfies with equality the constraint associated to the minimizer of (2.2).

A first technique to efficiently solve the projection subproblem consists of generalizing (the main ideas of) the separation algorithm without greatly increasing its computation time. This *cannot* be simply achieved by repeated separation: such a projection method would call the separation algorithm at least twice, or usually 3 or 4 times, i.e., it could become 3 or 4 times slower than the separation algorithm.[4]

We here give only one example of a successful generalization of a separation algorithm. In the context of a (very) classical robust LP [6, 15] the separation of a given $\mathbf{x} \in \mathbb{R}^n$ reduces to minimizing a difference $c_a - (\mathbf{a} + \widehat{\mathbf{a}})^\top\mathbf{x}$ over a set of nominal constraints $(\mathbf{a}, c_a) \in \mathcal{A}_{\texttt{nom}}$ and over all possible deviations $\widehat{\mathbf{a}}$ of the nominal coefficients $\mathbf{a}$. The projection subproblem (2.2) reduces to minimizing $\frac{c_a - (\mathbf{a}+\widehat{\mathbf{a}})^\top\mathbf{x}}{(\mathbf{a}+\widehat{\mathbf{a}})^\top\mathbf{d}}$ over the same $(\mathbf{a}, c_a)$ and over the same $\widehat{\mathbf{a}}$. Both subproblems can be solved by iterating over the nominal constraints $\mathcal{A}_{\texttt{nom}}$; for each $(\mathbf{a}, c_a) \in \mathcal{A}_{\texttt{nom}}$, the separation subproblem tries to

[4]A first call to the separation subproblem is needed to find a constraint satisfied with equality by some $\mathbf{x} + t_1\mathbf{d}$, followed by *at least* a second call to check if $\mathbf{x} + t_1\mathbf{d}$ can be separated. If $\mathbf{x} + t_1\mathbf{d}$ can be separated by a new constraint, one can find a solution $\mathbf{x} + t_2\mathbf{d}$ binding to this new constraint, with $t_2 < t_1$; the process could be repeated, leading to some $\mathbf{x} + t_3\mathbf{d}$ with $t_3 < t_2$, etc. Preliminary experiments suggest that a 3rd or a 4th call is needed at most iterations. It is more fruitful to explore techniques that can bring us close to designing a projection algorithm as fast as the separation one.

minimize the above difference while the projection subproblem tries to minimize the above ratio. This objective function change (minimize a ratio instead of a difference) does not change the nature of the subproblem algorithm: for both subproblems, the main computational bottleneck consists of iterating over $\mathcal{A}_{\texttt{nom}}$ [15, section 2.1].

A second technique to solve (2.2) is applicable to the (numerous) problems in which the constraints of $\mathscr{P}$ are associated to the feasible solutions of an auxiliary LP. This is the case for most Benders decomposition models (section 3.1) in which the separation subproblem is often formulated as an LP over a Benders subproblem polytope $\mathcal{P}$. In this case, (2.2) reduces to a linear-fractional program that can be reformulated as a pure LP using the Charnes–Cooper transformation [3]. This leads to an algorithm of the same complexity as the separation one, i.e., they both have the complexity of solving an LP over $\mathcal{P}$.

This technique can be generalized to the (numerous) problems in which the constraints of $\mathscr{P}$ are given by the feasible solutions of an integer LP (ILP). We will develop this idea in section 3.2, where $\mathscr{P}$ is the dual polytope of a `Column Generation` model for graph coloring. In this model, each constraint $(\mathbf{a}, c_a) = (\mathbf{a}, 1) \in \mathcal{A}$ of $\mathscr{P}$ is associated to a primal column, which, in turn, is given by the incidence vector $\mathbf{a} \in \{0,1\}^n$ of a stable in the considered graph. The stables of the graph can be seen as the integer solutions of a (stable set) polytope defined by edge inequalities. For this case, we propose a discrete Charnes–Cooper transformation that turns (2.2) into a disjunctive LP (DLP) and the integrality constraints $a_i \in \{0,1\}$ into disjunctive constraints of the form $\overline{a}_i \in \{0, \overline{\alpha}\}$, where $\overline{\alpha}$ is an additional decision variable. This DLP has a discrete feasible area and can be solved with the same techniques as the separation ILP, i.e., using a `Branch and Bound` with bounds determined from continuous relaxations. We will argue that there is no deep reason why such DLPs should be much harder in absolute terms than the ILP solved by the separation subproblem.[5]

We now give a last example of a technique for solving (2.2). When (2.2) is a dual LP in `Column Generation`, the separation subproblem can often be solved by `Dynamic Programming`, especially when the primal columns satisfy a resource consumption constraint (as in capacitated routing problems). In certain such cases, if the separation subproblem can be solved by `Dynamic Programming`, so can the projection subproblem. Recall that the main computational bottleneck in `Dynamic Programming` is to generate all states. If the number of states is similar for the separation and the projection, once all states are generated, it is not difficult to find the state of minimum objective value (either for a linear objective or for a fractional one). This is confirmed by the numerical experiments from the follow-up work [15, section 3.2].

3. Adapting the new method to different problems.

3.1. The Projective Cutting-Planes for the Benders reformulation.

3.1.1. The model with prohibitively many constraints and their separation. Introduced in the 1960s [2], the Benders method has become a widely used `Cutting-Planes` approach to solve (mixed-)integer linear programs of the form

$$\min\left\{\mathbf{b}^\top\mathbf{x} :\ \mathbf{B}\mathbf{x} + \mathbf{A}\mathbf{y} \geq \mathbf{c},\ \mathbf{x} \in \mathbb{Z}^n_+,\ \mathbf{y} \geq \mathbf{0}\right\}. \tag{3.1}$$

[5]The two programs have rather similar continuity-breaking constraints ($\overline{a}_i \in \{0, \overline{\alpha}\}$, respectively, $a_i \in \{0,1\}$), and they are typically solved with similar `Branch and Bound` methods that calculate bounds by lifting these continuity-breaking constraints. More exact details are provided in Remark 2 for graph coloring. We chose graph coloring only because it is a problem without complex problem–specific constraints whose presentation could impair the understanding of the more general ideas.

The goal is to minimize the cost $\mathbf{b}^\top \mathbf{x}$, while allowing the system of inequalities $\mathbf{Bx} + \mathbf{Ay} \geq \mathbf{c}$ to have a feasible solution $\mathbf{y} \geq \mathbf{0}$. The variables $\mathbf{y}$ could quantify flows in network design problems [5], goods delivered to customers in facility location problems, second-stage uncertain events in two-stage stochastic LPs, etc. The integrality $\mathbf{x} \in \mathbb{Z}_+^n$ can be lifted in certain problems or to calculate a lower bound.

Considering a fixed $\mathbf{x}$, the system of inequalities $\mathbf{Ay} \geq \mathbf{c} - \mathbf{Bx}$ admits a feasible solution $\mathbf{y}$ if and only if one can state $\min\left\{\mathbf{0}^\top \mathbf{y} : \ \mathbf{Ay} \geq \mathbf{c} - \mathbf{Bx}, \ \mathbf{y} \geq \mathbf{0}\right\} = 0$. Writing the dual of this LP, any dual feasible solution $\mathbf{u}$ has to belong to $\mathcal{P} = \left\{\mathbf{u} \geq \mathbf{0}_m : \mathbf{A}^\top \mathbf{u} \leq \mathbf{0}_n\right\}$, where m is the number of rows of $\mathbf{A}$. The dual objective value associated to $\mathbf{u}$ has to be nonpositive, i.e., we obtain $(\mathbf{c} - \mathbf{Bx})^\top \mathbf{u} \leq 0$. Referring to [14, section 2.1] or [5] for full details on this Benders reformulation process, (3.1) can be equivalently written in the following Benders decomposition form:

$$\left.\begin{aligned} &(3.2a) && \min \ \mathbf{b}^\top \mathbf{x} \\ &(3.2b) && \mathbf{c}^\top \mathbf{u} - (\mathbf{Bx})^\top \mathbf{u} \leq 0 \quad \forall \mathbf{u} \in \mathcal{P} \text{ s.t. } \mathbf{1}_m{}^\top \mathbf{u} = 1 \\ &(3.2c) && \mathbf{x} \in \mathbb{Z}_+^n, \end{aligned}\right\} \mathscr{P}$$

where $\mathcal{P}$ below is the *Benders subproblem polytope* that does not depend on $\mathbf{x}$:

$$\mathcal{P} = \left\{\mathbf{u} \geq \mathbf{0}_m : \mathbf{A}^\top \mathbf{u} \leq \mathbf{0}_n\right\}. \tag{3.3}$$

To solve this ILP, the Benders method applies a standard `Cutting-Planes` algorithm in which the outer approximations $\mathscr{P}_1 \supsetneq \mathscr{P}_2 \supsetneq \cdots \supset \mathscr{P}$ generated along the iterations are interpreted as discrete sets. At each iteration `it`, we say $\mathscr{P}_{\texttt{it}}$ corresponds to a relaxed master associated to (3.2a)–(3.2c), obtained by only keeping a subset of the constraints (3.2b). In fact, the only difference compared to the general large-scale (2.1) is that $\mathbf{x}$ is integer and $\texttt{opt}(\mathscr{P}_{\texttt{it}})$ needs to be determined using an ILP solver instead of an LP solver. Given the current optimal solution $\mathbf{x} = \texttt{opt}(\mathscr{P}_{\texttt{it}})$, the separation subproblem asks to solve the following LP to (try to) exclude $\mathbf{x}$ from $\mathscr{P}$:

$$\max\left\{\mathbf{c}^\top \mathbf{u} - (\mathbf{Bx})^\top \mathbf{u} : \ \mathbf{u} \in \mathcal{P}, \ \mathbf{1}_m{}^\top \mathbf{u} = 1\right\}. \tag{3.4}$$

The normalization $\mathbf{1}_m{}^\top \mathbf{u} = 1$ from (3.2b) and (3.4) can be considered superfluous in theory, but it is useful to avoid numerical issues in practice.[6]

3.1.2. Applying Projective Cutting-Planes. To solve (3.2a)–(3.2c) using `Projective Cutting-Planes`, one can proceed exactly as indicated in section 2, with only one exception: $\mathbf{x}$ is integer in (3.2a)–(3.2c), so that the master problem becomes an (NP-hard) ILP, i.e., one needs to call an ILP solver to determine $\texttt{opt}(\mathscr{P}_{\texttt{it}})$ at each iteration `it`. The iterative call to this ILP solver becomes the main computational bottleneck of the overall `Projective Cutting-Planes`. This is an encouraging factor for adopting the new method: the projection subproblem is an LP that can be solved *very rapidly* compared to the master problem which is an ILP.

Besides the above standard Benders reformulation (3.2a)–(3.2c), we will also study a linear relaxation that replaces $\mathbf{x} \in \mathbb{Z}_+^n$ with $\mathbf{x} \in \mathbb{R}_+^n$ in (3.2c). For both the integer

[6]This condition is superfluous because all positive multiples of $\mathbf{u} \in \mathcal{P}$ belong to $\mathcal{P}$ and they all produce the same inequality (3.2b), i.e., the status of this inequality does not change by multiplying all its terms by a positive constant. In practice, though, this normalization enables the separation algorithm to return only normalized constraints (3.2b), with no exceedingly large term.

and the relaxed models, a key question is the choice of the interior point $\mathbf{x}_{\mathtt{it}}$ at each iteration $\mathtt{it} \geq 1$. The first feasible solution $\mathbf{x}_1$ is determined by a very simple heuristic (section 3.1.4). For $\mathtt{it} > 1$, experiments suggest it is preferable to choose an interior point $\mathbf{x}_{\mathtt{it}}$ relatively far from the boundary. We usually set $\mathbf{x}_{\mathtt{it}} = \mathbf{x}_{\mathtt{it}-1} + \alpha t^*_{\mathtt{it}-1}\mathbf{d}_{\mathtt{it}-1}$ with $\alpha = 0.2$ except that we switch to a more aggressive $\alpha = 0.4$ in the integer model for $\mathtt{it} \geq 100$ (towards the end of the search).

Remark 1. Another consequence of the condition $\mathbf{x} \in \mathbb{Z}^n_+$ is that the projection algorithm might return a pierce point $\mathbf{x}_{\mathtt{it}} + t^*_{\mathtt{it}}\mathbf{d}_{\mathtt{it}}$ that is not an integer. However, for most problems, one can usually build such an integer feasible solution by simply rounding up all components of $\mathbf{x}_{\mathtt{it}} + t^*_{\mathtt{it}}\mathbf{d}_{\mathtt{it}}$. At least when $\mathbf{x}$ encodes design decisions to install (transmission) facilities, there is generally no reason to forbid an increase (by rounding) of the number of these facilities. This rounding-up has no impact on any algorithmic decision (like choosing a point $\mathbf{x}_{\mathtt{it}+1}$ that may remain noninteger), because it is only used when one needs a feasible integer point.

3.1.3. The projection subproblem algorithm. Consider an interior point $\mathbf{x}$ satisfying all constraints (3.2b) and a direction $\mathbf{d} \in \mathbb{R}^n$. Following Definition 2.1 (page 1010), the projection subproblem $\mathtt{project}(\mathbf{x} \to \mathbf{d})$ requires finding

(1) the maximum step length $t^* \geq 0$ such that $\mathbf{x} + t^*\mathbf{d}$ satisfies all constraints (3.2b);
(2) a vector $\mathbf{u} \in \mathcal{P}$ such that the constraint (3.2b) associated to $\mathbf{u}$ is respected with equality by the pierce point $\mathbf{x} + t^*\mathbf{d}$. This $\mathbf{u}$ may not be normalized: there is no need to multiply it by some factor to satisfy $\mathbf{1}_m{}^\top \mathbf{u} = 1$ like in (3.2b).

Substituting $\mathbf{x} + t^*\mathbf{d}$ for $\mathbf{x}$ into (3.2b), $\mathtt{project}(\mathbf{d} \to \mathbf{x})$ requires finding the maximum value t^* such that $\mathbf{c}^\top\mathbf{u} - (\mathbf{B}(\mathbf{x} + t^*\mathbf{d}))^\top \mathbf{u} \leq 0 \ \forall \mathbf{u} \in \mathcal{P}$, equivalent to $-t^*(\mathbf{Bd})^\top \mathbf{u} \leq (\mathbf{Bx})^\top \mathbf{u} - \mathbf{c}^\top\mathbf{u} \ \forall \mathbf{u} \in \mathcal{P}$. The right-hand side of this last inequality is always nonnegative because $\mathbf{x}$ is feasible and satisfies all constraints (3.2b). Furthermore, any $\mathbf{u} \in \mathcal{P}$ associated to a nonpositive $-(\mathbf{Bd})^\top \mathbf{u} \leq 0$ would allow t^* to be arbitrarily large. We can focus only on the $\mathbf{u} \in \mathcal{P}$ that satisfy $-(\mathbf{Bd})^\top \mathbf{u} > 0$, and so, t^* can be determined by solving the following linear-fractional program:

$$t^* = \min\left\{ \frac{(\mathbf{Bx} - \mathbf{c})^\top \mathbf{u}}{-(\mathbf{Bd})^\top \mathbf{u}} : \ \mathbf{u} \in \mathcal{P}, \ -(\mathbf{Bd})^\top \mathbf{u} > 0 \right\}. \tag{3.5}$$

This program can be translated to a standard LP using the Charnes–Cooper transformation [3]. Writing $\overline{\mathbf{u}} = \frac{\mathbf{u}}{-(\mathbf{Bd})^\top\mathbf{u}}$, we obtain $\mathbf{u} \in \mathcal{P} \implies \mathbf{A}^\top\overline{\mathbf{u}} \leq \mathbf{0}_n$, $\overline{\mathbf{u}} \geq \mathbf{0}_m$, $-(\mathbf{Bd})^\top \overline{\mathbf{u}} = 1$, and one can show that (3.5) is completely equivalent to

$$t^* = \min\left\{ (\mathbf{Bx} - \mathbf{c})^\top \overline{\mathbf{u}} : \ \mathbf{A}^\top\overline{\mathbf{u}} \leq \mathbf{0}_n, \ \overline{\mathbf{u}} \geq \mathbf{0}_m, \ -(\mathbf{Bd})^\top \overline{\mathbf{u}} = 1 \right\}. \tag{3.6}$$

It is not hard to check that the above change of variable $\mathbf{u} \to \overline{\mathbf{u}}$ transforms a feasible solution of (3.5) into a feasible solution of (3.6) with the same objective value and vice versa. The projection algorithm for (3.6) requires the same asymptotic running time as the separation algorithm that solves (3.4): both subproblems have the complexity of solving an LP with m variables and n or $n+1$ constraints.

3.1.4. From the general Benders model to a network design problem. So far, we have only discussed the general Benders model (3.2a)–(3.2c), avoiding describing any specific problem whose details could impair the understanding of the main ideas. However, we will provide numerical results on a network design problem [14, section 3.1] that asks to install multiple times a technology (e.g., cables of bandwidth b_{wd}) on the edges of a graph $G = (V, E)$. The installed links must be able to transfer data from a source or origin $O \in V$ towards a set of terminals $T \subsetneq V$, each $i \in T$ having a flow (data) demand of f_i. We use design variables $\mathbf{x} \in \mathbb{Z}_+^n$ to represent the number of links installed on each edge (so that $|E| = n$) and $\mathbf{y} \geq \mathbf{0}$ to encode data flows along edges.

The main Benders ILP (3.1) is instantiated as follows. The objective function minimizes the cost (number) of the installed links: $\min \sum_{\{i,j\}\in E} x_{ij} = \min \mathbf{1}_n^\top \mathbf{x}$. We impose (modified) flow conservation constraints $\sum_{\{i,j\}\in E} y_{ji} - \sum_{\{i,j\}\in E} y_{ij} \geq 0 \ \forall i \notin T \cup \{O\}$, $\sum_{\{i,j\}\in E} y_{ji} - \sum_{\{i,j\}\in E} y_{ij} \geq f_i \ \forall i \in T$, and bandwidth constraints $y_{ij} + y_{ji} \leq b_{\mathrm{wd}} x_{ij} \forall \{i,j\} \in E,\ i < j$. We also have $x_{ij} \in \mathbb{Z}_+, y_{ij}, y_{ji} \geq 0\ \forall \{i,j\} \in E,\ i < j$. One may also check [14, section 3.1] for full explanations on this model.

Referring to [14, section 3.2] for details on the intermediate steps of the Benders decomposition process, the final Benders reformulation model is given by (3.7a)–(3.7c) below, an instance of (3.2a)–(3.2c).

$$\left.\begin{array}{lc} \text{(3.7a)} & \min \mathbf{1}_n^\top \mathbf{x} \\ \text{(3.7b)} & \sum_{i\in T} f_i u_i - \sum_{\{i,j\}\in E} b_{\mathrm{wd}} x_{ij} u_{ij} \leq 0 \ \forall \mathbf{u} \in \mathcal{P} \text{ s.t. } \mathbf{1}^\top \mathbf{u} = 1 \\ \text{(3.7c)} & \mathbf{x} \in \mathbb{Z}_+^n, \end{array}\right\} \mathscr{P}$$

where $\mathcal{P}$ is described by (3.8) below.

$$\text{(3.8)} \qquad \mathcal{P} = \left\{\mathbf{u} \geq \mathbf{0} :\ -u_{ij} - u_i + u_j, -u_{ij} - u_j + u_i \leq 0 \ \ \forall \{i,j\} \in E,\ i < j\right\}.$$

The `Projective Cutting-Planes` for the general Benders model (3.2a)–(3.2c) from section 3.1.2 can be directly applied to solve (3.7a)–(3.7c). We construct the very first feasible solution $\mathbf{x}_1$ by assigning to each edge $\{i,j\} \in E$ the value $\left\lceil \frac{\sum_{i\in T} f_i}{b_{\mathrm{wd}}} \right\rceil$, so that each edge has enough capacity to transfer all the demands, making this $\mathbf{x}_1$ certainly feasible. The first direction is $\mathbf{d}_1 = -\mathbf{1}_n$, i.e., the direction with the fastest rate of objective function improvement.

To solve the projection subproblem `project`$(\mathbf{x} \to \mathbf{d})$ for some $\mathbf{x}$ that satisfies (3.7b), we will instantiate the general linear-fractional program (3.5), following the development from section 3.1.3. Accordingly, notice that the numerator of (3.5) contains a term $(\mathbf{Bx})^\top \mathbf{u}$ that was built from the terms involving $\mathbf{x}$ in the constraint (3.2b). Since (3.2b) has been instantiated to (3.7b), one can check that $(\mathbf{Bx})^\top \mathbf{u}$ corresponds to $\sum_{\{i,j\}\in E} b_{\mathrm{wd}} x_{ij} u_{ij}$. Using the fact that $\mathbf{d}$ is defined in the same space as $\mathbf{x}$, one can also check that $(\mathbf{Bd})^\top \mathbf{u}$ becomes $\sum_{\{i,j\}\in E} b_{\mathrm{wd}} d_{ij} u_{ij}$. Finally, $\mathbf{c}^\top \mathbf{u}$ represents the free terms (without $\mathbf{x}$) from (3.2b) that correspond to $\sum_{i\in T} f_i u_i$. We thus obtain that (3.5) is instantiated as

$$\text{(3.9)} \quad t^* = \min \left\{ \frac{\sum_{\{i,j\}\in E} b_{\mathrm{wd}} x_{ij} u_{ij} - \sum_{i\in T} f_i u_i}{-\sum_{\{i,j\}\in E} b_{\mathrm{wd}} d_{ij} u_{ij}} : \mathbf{u} \in \mathcal{P}, -\sum_{\{i,j\}\in E} b_{\mathrm{wd}} d_{ij} u_{ij} > 0 \right\}.$$

Recalling how we translated the linear-fractional program (3.5) to the LP (3.6), we apply the same Charnes–Cooper transformation to reformulate (3.9) as a pure LP.

As such, by substituting $\overline{\mathbf{u}} = \frac{\mathbf{u}}{-\sum_{\{i,j\}\in E} b_{\mathrm{wd}} d_{ij} u_{ij}}$, (3.9) is equivalent to[7]

$$\begin{aligned}
&(3.10a) && t^* = \min \ \textstyle\sum_{\{i,j\}\in E} b_{\mathrm{wd}} x_{ij} \overline{u}_{ij} - \sum_{i\in T} f_i \overline{u}_i \\
&(3.10b) && -\overline{u}_{ij} - \overline{u}_i + \overline{u}_j \le 0 \ \ \forall \{i,j\} \in E,\ i<j, \\
&(3.10c) && -\overline{u}_{ij} - \overline{u}_j + \overline{u}_i \le 0 \ \ \forall \{i,j\} \in E,\ i<j, \\
&(3.10d) && -\textstyle\sum_{\{i,j\}\in E} b_{\mathrm{wd}} d_{ij} \overline{u}_{ij} = 1, \\
&(3.10e) && \overline{\mathbf{u}} \ge \mathbf{0}.
\end{aligned}$$

3.2. The Projective Cutting-Planes for the graph coloring Column Generation model. In `Column Generation`, the generic LP (2.1) is instantiated as the dual of a relaxed master LP of the form $\min\{\sum c_a y_a : \sum a_i y_a \ge b_i \ \forall i \in [1..n]\}$, where all the sums are calculated over all columns $(\mathbf{a}, c_a) \in \mathcal{A}$. These columns may encode stables in graph coloring, cutting patterns in *(Multiple-Length) Cutting-Stock*, routes in vehicle routing problems, assignments of courses to timeslots in timetabling, or any other specific subsets in the most general set-covering problem. Referring the reader to [13, 1, 17, 11] for full details, let us focus on the dual LP:

$$(3.11) \qquad y_a : \left.\begin{array}{ll} \max \mathbf{b}^\top \mathbf{x} & \\ \mathbf{a}^\top \mathbf{x} \le c_a & \forall (\mathbf{a}, c_a) \in \mathcal{A} \\ \mathbf{x} \ge \mathbf{0}_n. & \end{array}\right\} \mathscr{P}$$

The standard `Column Generation` can be seen as a `Cutting-Planes` algorithm (e.g., Kelley's method) acting on this dual LP (3.11). Besides graph coloring, (3.11) will also serve to model *Multiple-Length Cutting-Stock* in the follow-up paper [15].

3.2.1. The separation and the standard Cutting-Planes. Graph coloring is a set covering problem that asks to determine the minimum number of stables of a given graph $G(V,E)$ needed to cover (color) each vertex of V once. Focusing on (3.11), each constraint $(\mathbf{a}, c_a) \in \mathcal{A}$ corresponds to the incidence vector $\mathbf{a}$ of a stable of G; we assume $c_a = 1$ (each color counts once) and $\mathbf{b} = \mathbf{1}_n$ (each vertex has to receive one color). Such assumptions may differ in other graph coloring variants; e.g., in multicoloring $\mathbf{b}$ is different from $\mathbf{1}_n$.

To solve (3.11) by `Column Generation`, one needs to solve at each iteration `it` the separation subproblem $\min_{(\mathbf{a},1)\in\mathcal{A}} 1 - \mathbf{a}^\top \mathbf{x}$, where $\mathbf{x}$ is the current optimal (outer) solution $\texttt{opt}(\mathscr{P}_{\texttt{it}})$. In standard graph coloring, the constraints $\mathcal{A}$ are given by the stables of G, and the separation subproblem reduces to the maximum weight stable problem with weights $\mathbf{x}$ which is NP-hard. The `Column Generation` formulation of graph coloring has been widely studied (see [12, 9] and references therein); aside from popularity reasons, this also comes from the fact that graph coloring is a rather generic problem with simple constraints. There seems to be less potential in analyzing or exploiting such constraints than in developing more general optimization techniques. According to the abstract of [9], `Column Generation` is also the "best method known for determining lower bounds on the vertex coloring number."

3.2.2. The Projective Cutting-Planes for graph coloring. We need very few customizations to apply the generic `Projective Cutting-Planes` from section 2 on (3.11). First, we define $\mathbf{x}_{\texttt{it}} = \mathbf{x}_{\texttt{it}-1} + t^*_{\texttt{it}-1}\mathbf{d}_{\texttt{it}-1}$ at each iteration $\texttt{it} > 1$, so that $\mathbf{x}_{\texttt{it}}$ becomes the best feasible solution found so far (the last pierce point); graph

[7]In both (3.10b)–(3.10c) and (3.8), we use the convention that if i (resp., j) is the source then the term u_i (resp., u_j) vanishes, as in footnote 6 of [14].

coloring is the only problem from this study for which this choice leads to good results in the long run. For `it` $= 1$, we simply take $\mathbf{x}_1 = \mathbf{0}_n$; $\mathbf{d}_1$ is chosen to point towards a direction obtained from an initial feasible coloring found by a heuristic (see footnote 14, page 1025), i.e., to construct $\mathbf{d}_1$, we assign to each $v \in V = [1..n]$ the value $\frac{1}{|\texttt{stab}(v)|}$, where $\texttt{stab}(v) \ni v$ is the stable containing v in the given feasible coloring. Using such initial feasible coloring offers multiple advantages, both for `Projective Cutting-Planes` and for the standard `Column Generation`.

– This initial coloring provides a set of stables or initial constraints $\mathcal{A}_0$ in (3.11), so as to start from the very first iteration with a reasonable outer approximation $\mathscr{P}_0 \supsetneq \mathscr{P}$, i.e., the overall solution process is warm-started.
– The first outer approximation $\mathscr{P}_0$ obtained as above leads to a first upper bound $\mathbf{b}^\top \texttt{opt}(\mathscr{P}_0)$ that is equal to the number of colors used by the heuristic coloring. This upper bound is used as a marker to evaluate the gap of all lower bounds reported along all iterations. Without this initial coloring, one might need dozens or hundreds of iterations to obtain an upper bound of the same quality.
– If we had started by projecting $\mathbf{0}_n \to \mathbf{1}_n$, we would have obtained a very first pierce point $t_1^*\mathbf{1}_n = \frac{1}{\alpha(G)}\mathbf{1}_n$ that might correspond to multiple constraints of (3.11) associated to multiple stables of maximum size $\alpha(G)$. If one then takes $\mathbf{x}_2 = \mathbf{x}_1 + t_1^*\mathbf{1}_n = \frac{1}{\alpha(G)}\mathbf{1}_n$, the second projection can return $t_2^* = 0$ because of a second stable of size $\alpha(G)$. If this repeats a third or a fourth time, the iterative solution process could stall for too many iterations, generating a form of degeneracy.

We will also use (in section 3.2.4) a second coloring model in which we define the constraints $\mathcal{A}$ using a new (broader) notion of reinforced relaxed stables. In this new model, each element of $\mathcal{A}$ is associated to a solution of an auxiliary polytope $\mathcal{P}$ that contains the standard stables. The disadvantage is that $\mathcal{P}$ contains many other elements besides legitimate stables so that the new (3.11) model contains more constraints. But the advantage is that the projection subproblem becomes considerably simpler as it can be formulated as a pure LP (section 3.2.4). The `Projective Cutting-Planes` described above will be applied in the same manner, but it will generate faster and still high-quality lower bounds.

3.2.3. The projection subproblem. Instantiating (2.2) on standard graph coloring, the projection subproblem $\texttt{project}(\mathbf{x} \to \mathbf{d})$ becomes

$$t^* = \min_{\mathbf{a}} \frac{1 - \mathbf{x}^\top \mathbf{a}}{\mathbf{d}^\top \mathbf{a}}, \tag{3.12a}$$

$$\mathbf{d}^\top \mathbf{a} > 0, \tag{3.12b}$$

$$\mathcal{P}_{0-1} \left\{ \begin{array}{ll} a_i + a_j \leq 1 & \forall \{i,j\} \in E, \\ a_i \in \{0,1\} & \forall i \in [1..n]. \end{array} \right. \tag{3.12c}$$

We will translate this integer linear-fractional program to a disjunctive LP (DLP), transforming the integrality constraints $a_i \in \{0,1\}$ into disjunctive constraints of the form $\overline{a}_i \in \{0, \overline{\alpha}\}$ $\forall i \in [1..n]$. We will see that a disjunctive constraint breaks the continuity in the same manner as an integrality constraint; thus, the DLP has a discrete feasible area and can be solved with similar `Branch and Bound` methods as an ILP. We recall that the standard separation subproblem is the standard ILP $\min\left\{1 - \mathbf{x}^\top \mathbf{a} : \ \mathbf{a} \in \mathcal{P}_{0-1}\right\}$. The constraints $a_i + a_j \leq 1 \forall \{i,j\} \in E$ from (3.12.c) are referred to as edge inequalities [10, 12]; $\mathcal{P}_{0-1}$ is the set of standard stables, and its convex closure $\texttt{conv}(\mathcal{P}_{0-1})$ is referred to as the stable set polytope.

To write (3.12.a)–(3.12.c) as a DLP, we propose applying a discrete version of the Charnes–Cooper transformation initially applied for standard LPs [3]. Accordingly, let us consider a change of variables $\overline{\mathbf{a}} = \frac{\mathbf{a}}{\mathbf{d}^\top \mathbf{a}}$ and $\overline{\alpha} = \frac{1}{\mathbf{d}^\top \mathbf{a}}$; we will prove that (3.12.a)–(3.12.c) is completely equivalent to

$$t^* = \min_{\overline{\mathbf{a}},\overline{\alpha}} \overline{\alpha} - \mathbf{x}^\top \overline{\mathbf{a}}, \tag{3.13a}$$

$$\overline{a}_i + \overline{a}_j \leq \overline{\alpha} \qquad \forall \{i,j\} \in E, \tag{3.13b}$$

$$\mathbf{d}^\top \overline{\mathbf{a}} = 1, \tag{3.13c}$$

$$\overline{a}_i \in \{0, \overline{\alpha}\} \qquad \forall i \in [1..n], \tag{3.13d}$$

$$\overline{\alpha} \geq 0. \tag{3.13e}$$

To prove this equivalence, let us first show that the change of variables $\mathbf{a} \to \overline{\mathbf{a}}, \overline{\alpha}$ maps a feasible solution $\mathbf{a}$ of (3.12.a)–(3.12.c) to a feasible solution of (3.13.a)–(3.13.e) with the same objective value. For this, we divide each term of (3.12.b)–(3.12.c) by $\mathbf{d}^\top \mathbf{a} > 0$; after replacing $\overline{a}_i = \frac{a_i}{\mathbf{d}^\top \mathbf{a}}$ $\forall i \in [1..n]$ and $\overline{\alpha} = \frac{1}{\mathbf{d}^\top \mathbf{a}} > 0$, we obtain the constraints of the DLP. Conversely, a feasible solution of (3.13.a)–(3.13.e) can be reversely mapped to $\mathbf{a} = \overline{\alpha}^{-1} \cdot \overline{\mathbf{a}}$ to obtain a feasible solution $\mathbf{a}$ in (3.12.a)–(3.12.c). Note that $\mathbf{a} = \overline{\alpha}^{-1} \cdot \overline{\mathbf{a}}$ is consistent because $\overline{\alpha} = 0$ would lead to $\overline{\mathbf{a}} = \mathbf{0}_n$ via (3.13.d), rendering (3.13.c) infeasible. One can directly check that the resulting $\mathbf{a}$ satisfies all constraints in the initial program. The equality of the objective values follows from $\overline{\alpha} - \mathbf{x}^\top \overline{\mathbf{a}} = \frac{1}{\mathbf{d}^\top \mathbf{a}} - \frac{\mathbf{x}^\top \overline{\mathbf{a}}}{\mathbf{d}^\top \overline{\mathbf{a}}} = \frac{1 - \mathbf{x}^\top \mathbf{a}}{\mathbf{d}^\top \mathbf{a}}$.

Remark 2. The DLP (3.13.a)–(3.13.e) uses disjunctive constraints $\overline{a}_i \in \{0, \overline{\alpha}\}$ instead of the integrality constraints $a_i \in \{0,1\}$ of the separation subproblem ILP $\min \left\{1 - \mathbf{a}^\top \mathbf{x} : a_i + a_j \leq 1 \ \forall \{i,j\} \in E,\ \mathbf{a} \in \mathbb{Z}^n_+\right\}$. Both the projection DLP and the separation ILP can be solved with similar `Branch and Bound` methods that calculate lower bounds by lifting the disjunctive or, respectively, integrality constraints. These constraints break the continuity in a similar manner, and we find *no* deep theoretical difference that would make one much easier to handle than the other.

Remark 3. Despite the above theoretical similarity between the DLP and the ILP, the practical situation may be different as of 2019. We solve both the DLP and the ILP with `cplex`, implementing the disjunctive constraints as logical constraints. As such, we implicitly use a larger arsenal on the ILP. For instance, `cplex` applies many valid inequalities on the ILP, but it does not "realize" that such ILP cuts could be translated to DLP valid inequalities.[8] This comes from the fact that `cplex` does *not* have the notion of valid inequalities satisfied by all "discrete" DLP solutions that satisfy $\overline{a}_i \in \{0, \overline{\alpha}\}$, $i \in [1..n]$; it does not see $\overline{a}_i \in \{0, \overline{\alpha}\}$ as somehow similar to an integrality constraint (in the sense that $\frac{\overline{a}_i}{\overline{\alpha}}$ is integer). In fact, it does not even have the information that lifting a disjunctive constraint $\overline{a}_i \in \{0, \overline{\alpha}\}$ enables $\overline{a}_i$ to take a fractional value between two "discrete" bounds 0 and $\overline{\alpha}$. As such, `cplex` cannot exploit this to generate refined branching rules as it does for the ILP.

For a fair comparison, the ILP algorithm should not be more elaborate than the DLP one. One option would be to disable all `cplex` options for cuts, branching, or

[8]The logs show that `cplex` uses many "zero-half cuts" on the ILP. According to the documentation, these cuts are simply "based on the observation that when the left-hand side of an inequality consists of integral variables and integral coefficients, then the right-hand side can be rounded down to produce a zero-half cut." For instance, it can sum up different constraints to obtain $a_1 + a_2 \leq 3.5$ which is reduced to the zero-half cut $a_1 + a_2 \leq 3$. The same operations could apply perfectly well on a disjunctive LP and $\overline{a}_1 + \overline{a}_2 \leq 3.5\overline{\alpha}$ could be reduced to $\overline{a}_1 + \overline{a}_2 \leq 3\overline{\alpha}$.

heuristics. But instead of reducing the strength of the ILP algorithm, we prefer to push the DLP algorithm to a higher level, striving to obtain competitive results in the end. For this purpose, we reinforce the DLP (3.13.a)–(3.13.e) by inserting k-clique inequalities (with $k = 4$) at the root node of the branching tree, when all disjunctive constraints (3.13.d) are lifted and the problem reduces to a pure LP. This LP is solved by a `cut generation` method described in section 3.2.4, searching at each iteration to separate the current solution using a k-clique inequality of the form $\sum_{i\in\mathscr{C}} a_i \leq 1$ (in which $\mathscr{C}$ is a k-clique), equivalent to $\sum_{i\in\mathscr{C}} \overline{a}_i \leq \overline{\alpha}$ in the DLP (3.13.a)–(3.13.e).

Most ideas above could naturally be extended to address more (diverse) constraints besides the edge inequalities $a_i + a_j \leq 1\ \forall\{i,j\} \in E$. If we replaced these edge inequalities with other linear constraints, the transformation (3.12.a)–(3.12.c) $\to$ (3.13.a)–(3.13.e) would work in a similar manner, i.e., any linear constraint can be reformulated using the Charnes–Cooper transformation. This suggests that the proposed approach could be applied on (numerous) `Column Generation` models in which the columns $\mathcal{A}$ represent the solutions of an ILP. In such cases, the separation subproblem is an ILP, and the projection subproblem can be written as a DLP, replacing $a_i \in \{0,1\}$ with $\overline{a}_i \in \{0, \overline{\alpha}\}$. In fact, we focused on standard graph coloring mainly because it is a problem with *no* particularly skewed constraints whose presentation could clutter (the design of) `Projective Cutting-Planes`.[9]

3.2.4. The projection subproblem in a second coloring model with RR-stables.

We here use a new coloring model that defines the (dual) constraints $\mathcal{A}$ using a broader notion of reinforced-relaxed stables (RR-stables).

DEFINITION 3.1. *A reinforced-relaxed stable (RR-stable) is a feasible solution of an auxiliary polytope $\mathcal{P}$ that represents an outer approximation of the stable set polytope $\texttt{conv}(\mathcal{P}_{0-1})$ from* (3.12.c). *We formally say $(\mathbf{a}, 1) \in \mathcal{A} \iff \mathbf{a} \in \mathcal{P}$, defining $\mathcal{P} \supsetneq \texttt{conv}(\mathcal{P}_{0-1})$ using six cut classes $\mathcal{R}$, as indicated below.*

$$\mathcal{P} = \left\{\mathbf{a} \geq \mathbf{0}_n :\ \mathbf{e}^\top\mathbf{a} \leq 1\ \forall(\mathbf{e}, 1) \in \mathcal{R},\ \mathbf{f}^\top\mathbf{a} \leq 0\ \forall(\mathbf{f}, 0) \in \mathcal{R}\right\}. \tag{3.14}$$

The set $\mathcal{R}$ contains six classes (a)–(f) *of reinforcing cuts whose design has been inspired by research in valid inequalities for the maximum stable problem* [10]. *The cuts* (a)–(d) *can all be enumerated, but the cuts* (e)–(f) *can be extremely numerous, and they are generated only when needed, using a separation subproblem. We present all these cut classes* (a)–(f) *below, but their exact implementation requires more engineering detail, and we also refer the reader to* [15, App. A.3.1] *for full descriptions.*

(a) *We first introduce all edge inequalities $a_u + a_v \leq 1\ \forall\{u,v\} \in E$ that, if used alone, would determine simple relaxed stables.*
(b) *We impose a clique inequality $\sum_{v\in\mathscr{C}} a_v \leq 1$ for each clique $\mathscr{C}$ such that $|\mathscr{C}| \leq$ $\min(5,k)$, where k is a parameter that defines the model; see point* (f) *below.*
(c) *Cuts* (c) *are all generated by constructing a family of cliques that cover all elements of V multiple times using an iterative routine* [15, App. A.3.1]*; each clique $\mathscr{C}$ constructed this way leads to a cut $\sum_{v\in\mathscr{C}} a_v \leq 1$.*
(d) *We generate a cut of this class for any $u, v, w \in V$ such that $\{u,v\} \in E$, $\{u,w\} \notin E$, and $\{v,w\} \notin E$. Writing $N_v = \{v' \in V : \{v,v'\} \in E\}$, we obtain the cut*

[9] For example, the defective coloring problem allows each vertex to have a maximum number of $d \geq 0$ neighbors of the same color. When $d = 0$, we obtain standard graph coloring. For $d > 0$, the edge inequalities evolve to $n(a_i - 1) + \sum_{\{i,j\}\in E} a_j \leq d\,\forall i \in [1..n]$. Note that this constraint is only active for $a_i = 1$. Applying the discrete Charnes–Cooper transformation, this constraint is translated to $n(\overline{a}_i - \overline{\alpha}) + \sum_{\{i,j\}\in E} \overline{a}_j \leq d\overline{\alpha}\ \forall i \in [1..n]$, obtaining a defecting coloring version of (3.13.b).

$a_u + a_v \le a_w + \mathbf{a}(N_w - N_u \cap N_v)$, where we used notation $\mathbf{a}(S) = \sum_{s \in S} a_s \ \forall S \subseteq V$. This idea has also been generalized to the case of triangles $\{\mu, u, v\} \subset V$ not connected to a vertex $w \in V$ [15, App. A.3.1].

(e) *We here consider odd-cycle inequalities $\sum_{v \in H} a_h \le \frac{|H|-1}{2}$, where H is an odd cycle. These cuts cannot all be generated: they are added by repeated separation.*

(f) *The last cut class consists of k-clique inequalities $\mathbf{a}(\mathscr{C}) \le 1$ associated to cliques $\mathscr{C}$ with at maximum k elements, where k is a parameter of the model. For large values of k, (the iterative call to) the separation subproblem for these cuts can become the main computational bottleneck of the overall `Cutting-Planes`. This is why we present in* [15, App. A.3.3] *a specific `Branch & Bound with Bounded Size` method devoted to this maximum weight clique problem with bounded size.*

Instantiating (2.2), the projection subproblem `project`$(\mathbf{x} \to \mathbf{d})$ becomes

$$t^* = \min \left\{ \frac{1 - \mathbf{x}^\top \mathbf{a}}{\mathbf{d}^\top \mathbf{a}} : \ \mathbf{a} \in \mathcal{P}, \ \mathbf{d}^\top \mathbf{a} > 0 \right\}. \tag{3.15}$$

This projection subproblem (3.15) is a linear-fractional program that can be translated to a standard LP using the Charnes–Cooper transformation [3]. More exactly, writing $\overline{\mathbf{a}} = \frac{\mathbf{a}}{\mathbf{d}^\top \mathbf{a}}$ and $\overline{\alpha} = \frac{1}{\mathbf{d}^\top \mathbf{a}}$, one can show that (3.15) is is equivalent to

$$t^* = \min \ \overline{\alpha} - \mathbf{x}^\top \overline{\mathbf{a}}, \tag{3.16a}$$

$$\mathbf{e}^\top \overline{\mathbf{a}} \le \overline{\alpha}, \ \mathbf{f}^\top \overline{\mathbf{a}} \le 0 \quad \forall (\mathbf{e}, 1) \in \mathcal{R}, \ \forall (\mathbf{f}, 0) \in \mathcal{R}, \tag{3.16b}$$

$$\mathbf{d}^\top \overline{\mathbf{a}} = 1, \tag{3.16c}$$

$$\overline{\mathbf{a}} \ge \mathbf{0}_n, \ \overline{\alpha} \ge 0. \tag{3.16d}$$

To prove this equivalence, we can follow the proof of the equivalence between (3.12.a)–(3.12.c) and (3.13.a)–(3.13.e) from section 3.2.3. The only difference is that we no longer have disjunctive constraints $\overline{a}_i \in \{0, \overline{\alpha}\}$ in (3.16a)–(3.16d); without these constraints, we can no longer show $\overline{\alpha} = 0 \implies \overline{\mathbf{a}} = \mathbf{0}_n$. However, this Charnes–Cooper transformation is general and works for any constraints (3.16b).[10]

The LP (3.16a)–(3.16d) is solved by `cut generation`. In fact, the cuts (a)–(d) are all generated at the first iteration, and only the cuts (e)–(f) are dynamically generated by solving the separation subproblem $\max \left(\max_{(\mathbf{e},1) \in \mathcal{R}} \mathbf{e}^\top \overline{\mathbf{a}} - \overline{\alpha}, \max_{(\mathbf{f},0) \in \mathcal{R}} \mathbf{f}^\top \overline{\mathbf{a}} \right)$ for each current optimal solution $(\overline{\mathbf{a}}, \overline{\alpha})$ [15, App. A.3.2]. Note that no cut in $\mathcal{R}$ depends on $\mathbf{x}$ or $\mathbf{d}$; as such, each cut generated at some iteration `it` of the overall `Projective Cutting-Planes` is kept/reused at all next iterations `it` $+1$, `it` $+2$, etc.

To make `Projective Cutting-Planes` reach its full potential, it is important to have a fast separation algorithm for the cuts (f) since the cuts (e) can be separated in polynomial time by applying Dijkstra's algorithm on a bipartite graph with $2n+2$ vertices (see point (e) in [15, App. A.3.1]). The difficulty of separating cuts (f) depends on the value of k, and we designed a specific `Branch & Bound with Bounded Size` algorithm that can be very fast when k is low; even for intermediate values of k, this

[10]For any constraints (3.16b), any feasible solution $(\overline{\mathbf{a}}, \overline{\alpha})$ with $\overline{\alpha} = 0$ of the LP (3.16a)–(3.16d) can always be associated to an extreme ray of feasible solutions in the initial linear-fractional program. More exactly, one can take any $\mathbf{a} \in \mathcal{P}$ and construct a ray $\mathbf{a} + z\overline{\mathbf{a}}$ of $\mathcal{P}$, i.e., $\mathbf{a} + z\overline{\mathbf{a}}$ is feasible in (3.14) for all $z \ge 0$. To check this, notice that $\mathbf{e}^\top(\mathbf{a} + z\overline{\mathbf{a}}) \le 1 \ \forall (\mathbf{e}, 1) \in \mathcal{R}$ follows from $\mathbf{a} \in \mathcal{P}$ and $\mathbf{e}^\top \overline{\mathbf{a}} \le 0 \ \forall (\mathbf{e}, 1) \in \mathcal{R}$, which holds because $\overline{\alpha} = 0$ in (3.16b); a similar argument proves $\mathbf{f}^\top \mathbf{a} \le 0 \ \forall (\mathbf{f}, 0) \in \mathcal{R}$. The objective value of $\mathbf{a} + z\overline{\mathbf{a}}$ in (3.15) converges to $\lim_{z \to \infty} \frac{1 - \mathbf{x}^\top \mathbf{a} - z\mathbf{x}^\top \overline{\mathbf{a}}}{\mathbf{d}^\top \mathbf{a} + z\mathbf{d}^\top \overline{\mathbf{a}}} = \lim_{z \to \infty} \frac{-z\mathbf{x}^\top \overline{\mathbf{a}}}{\mathbf{d}^\top \mathbf{a} + z} = -\mathbf{x}^\top \overline{\mathbf{a}}$.

algorithm can be much faster than existing state-of-the-art software for the (clique) problem with no size restriction (for $k = \infty$). The value of k controls a trade-off between computation time (for the overall the `Projective Cutting-Planes`) and the reported optimal value. When k is too small, the outer approximation $\mathcal{P} \supsetneq \texttt{conv}(\mathcal{P}_{0-1})$ from (3.14) may become too coarse: $\mathcal{P}$ may contain too many elements besides legitimate stables, leading to too many artificial constraints in the new (3.11) model with RR-stables, so that the lower bound reported in the end becomes smaller.

Remark 4. The optimum of the `Column Generation` model (3.11) with RR-stables can exceed the maximum clique size ω because cuts (d) can exclude $\left[\frac{1}{\omega}\ \frac{1}{\omega} \ldots \frac{1}{\omega}\right]^\top$ from $\mathcal{P}$. As such, the new model (3.11) does not necessarily contain a constraint of the form $\left[\frac{1}{\omega}\ \frac{1}{\omega} \ldots \frac{1}{\omega}\right]\mathbf{x} \le 1$, and so, the dual objective function value does not necessarily satisfy $\mathbf{1}_n^\top \mathbf{x} \le \omega$. More generally, the lower bounds $\mathbf{b}^\top \mathbf{x}_{\texttt{it}}$ of `Projective Cutting-Planes` remain perfectly valid for any (dual) objective function $\mathbf{b} \ne \mathbf{1}_n$ (e.g., in graph multicoloring), while ω is no longer a valid lower bound when $\mathbf{b} \ne \mathbf{1}_n$.

3.2.5. Projecting a boundary point $\mathbf{x}_{\texttt{it}}$ can lead to a null step-length $t^*_{\texttt{it}}$ in the model with RR-stables. For the model with RR-stables, choosing $\mathbf{x}_{\texttt{it}} = \mathbf{x}_{\texttt{it}-1} + t^*_{\texttt{it}-1}\mathbf{d}_{\texttt{it}-1}$ is not very effective because such an $\mathbf{x}_{\texttt{it}}$ is a boundary point that can belong to multiple facets. Solving the projection subproblem on such a point may (repeatedly) return a new facet that touches $\mathbf{x}_{\texttt{it}}$ and a zero step-length. This could make `Projective Cutting-Planes` stagnate like a Simplex algorithm that performs degenerate steps without improving the objective value.

More technically, this can be explained as follows. First, notice that $\mathbf{x}_{\texttt{it}} = \mathbf{x}_{\texttt{it}-1} + t^*_{\texttt{it}-1}\mathbf{d}_{\texttt{it}-1}$ belongs to the first-hit constraint/facet $\mathbf{a}^\top\mathbf{x} \le 1$ returned by the projection subproblem at iteration $\texttt{it}-1$, so that $\mathbf{a}^\top\mathbf{x}_{\texttt{it}} = 1$. Furthermore, the current optimal outer solution $\texttt{opt}(\mathscr{P}_{\texttt{it}-1})$ also belongs to the above first-hit facet, and so, by taking $\mathbf{d}_{\texttt{it}} = \texttt{opt}(\mathscr{P}_{\texttt{it}-1}) - \mathbf{x}_{\texttt{it}}$ as indicated by Step 2 from section 2, we also obtain $\mathbf{a}^\top\mathbf{d}_{\texttt{it}} = 0$. Now recall that $\mathbf{a}$ can be seen as a feasible solution (an RR-stable) of the polytope $\mathcal{P}$ from (3.14), so that there might exist a continuous set of RR-stables $\widehat{\mathbf{a}} \in \mathcal{P}$ very close to $\mathbf{a}$. There might exist multiple first-hit facets (of $\mathscr{P}$) that touch $\mathbf{x}_{\texttt{it}}$ because some of these $\widehat{\mathbf{a}} \in \mathcal{P}$ can satisfy $\widehat{\mathbf{a}}^\top\mathbf{x}_{\texttt{it}} = 1$; recall that $\mathbf{a} = \frac{\overline{\mathbf{a}}}{\overline{\alpha}}$ is *not* an extreme solution determined by optimizing the LP (3.16a)–(3.16d) in the direction of $\mathbf{x}_{\texttt{it}}$. As such, it is often possible to find some $\widehat{\mathbf{a}} \in \mathcal{P}$ such that $\widehat{\mathbf{a}}^\top\mathbf{x}_{\texttt{it}} = 1$ and $\widehat{\mathbf{a}}^\top\mathbf{d} = \epsilon > 0$, which leads to a step-length of $t^*_{\texttt{it}} = \frac{1-\widehat{\mathbf{a}}^\top\mathbf{x}_{\texttt{it}}}{\widehat{\mathbf{a}}^\top\mathbf{d}_{\texttt{it}}} = \frac{0}{\epsilon} = 0$.

We thus propose defining $\mathbf{x}_{\texttt{it}}$ as a strictly interior point using $\mathbf{x}_{\texttt{it}} = \alpha(\mathbf{x}_{i-1} + t^*_{i-1}\mathbf{d}_{i-1})$ with $\alpha = 0.9999$. This way, the above RR-stables $\widehat{\mathbf{a}}$ very close to $\mathbf{a}$ no longer cause any problem: they lead to $1 - \widehat{\mathbf{a}}^\top\mathbf{x}_{\texttt{it}} > 0$ and to a small ("ϵ-sized") $\widehat{\mathbf{a}}^\top\mathbf{d}_{\texttt{it}}$, so that $\frac{1-\widehat{\mathbf{a}}^\top\mathbf{x}_{\texttt{it}}}{\widehat{\mathbf{a}}^\top\mathbf{d}_{\texttt{it}}}$ becomes very large. Generally speaking, this 0.9999 factor is reminiscent of the "fraction-to-the-boundary stepsize factor" used in (some) interior point algorithms to prevent them from touching the boundary; see the parameter $\alpha_0 = 0.99$ in the pseudocode above section 3 in [7].

4. Numerical experiments.

We here evaluate the potential of `Projective Cutting-Planes` in a Benders reformulation context and then for graph coloring.[11]

[11]The `C++` source code is available online at http://cedric.cnam.fr/~porumbed/projcutplanes/. We compiled all programs with `g++ -O3`; we used `Cplex` 12.6 to solve all (integer) linear programs, using the concert technology for `C++`. All reported results were obtained on a mainstream `Linux` computer with a `i7-5500U CPU` and 16GB of RAM; unless otherwise stated, all programs use a single thread. There are 13279 lines altogether, including the experiments from [15].

4.1. The Benders reformulation. We consider the network design problem from section 3.1.4 as formalized by (3.7a)–(3.7c). We use a test bed of 14 existing instances from [14] and 7 new instances denoted `a`, `b`, ..., `g`; their characteristics are described in the first 5 columns of Table 1. The bandwidth is always fixed to $b_{\mathrm{wd}} = 3$, and all demands have been generated uniformly at random from an interval $[0, \mathtt{dem}_{\mathtt{max}}]$.

Table 1 compares the new and the standard methods on two problem variants:

1. The linear relaxation of (3.7a)–(3.7c) in the first group of 21 rows, i.e., we relax $\mathbf{x} \in \mathbb{Z}_+^n$ to $\mathbf{x} \in \mathbb{R}_+^n$ in (3.7c); it is useful to have a lower bound on the original ILP.
2. The original integer Benders model (3.7a)–(3.7c) in the second group of 10 rows.

The first 5 columns of Table 1 describe the instance: the instance class in column 1, the instance ID (number) in column 2, the number of edges $n = |E|$ in column 3, the number of vertices in column 4, and the maximum demand $\mathtt{dem}_{\mathtt{max}}$ in column 5. Column 6 is the optimum value. The columns *avg (std) min* report statistics over 10 runs on the number of iterations and on the total CPU time.[12] The columns *Time solve master* present the percentage of CPU time spent on solving master problems along the iterations, i.e., to calculate $\mathtt{opt}(\mathscr{P}_1)$, $\mathtt{opt}(\mathscr{P}_2)$, etc. For the linear relaxation only, column 7 (Best IP Sol) reports the best integer solution that `Projective Cutting-Planes` could determine along all runs using Remark 1 (page 1015).

4.1.1. The results on the linear relaxation. In the first 21 rows, the *average* number of iterations of `Projective Cutting-Planes` is often *better* than the best number of iterations of the standard `Cutting-Planes`, so that there is no need for an in-depth statistical test to confirm the difference. The new method can roughly reduce the average number of iterations by a factor of almost 3 (last `rnd-100` instance), or by a factor of about 2 for roughly a third of the instances. This speed-up is also superior to the one obtained by the best stabilized enhanced `Cutting-Planes` from [14].[13]

Columns 6 and 7 of the first 21 rows of Table 1 actually represent a lower bound and an upper bound for the integer optimum. The gap between these two bounds is 5.5% on average, or even below 1% if we restrict our attention to the instances with $\mathtt{dem}_{\mathtt{max}} \in \{100, 300\}$. This shows that the integration of `Projective Cutting-Planes` in a `Branch and Bound` for the integer model seems promising: the above gap could actually represent the gap at the root node of the branching tree, and it is well known that this root node gap has a high impact on the effectiveness of a `Branch and Bound`.

4.1.2. The results on the integer model. The last group of 10 rows concerns the smallest Benders instances from Table 1, because we solve the integer problem, which is far more difficult than the linear relaxation. Comparing columns 7 and 14 (labelled *avg*), the new method could reduce the number of iterations by a factor of 3 or 4 (rows `b` or `d`) or 2 (ante-penultimate row). The new method can also halve the average running time on four instances out of ten (see rows 2, 3, 4, and 7), although for some rare cases (two instances) it can also fail because of the numerical difficulties described in [15, App. A.2]. The notation $^{-\kappa}$ in columns 7 and 14 indicates that κ

[12]By default, the Benders algorithms from section 3.1 have no random component. However, we could randomize them by inserting 10 random cut-set constraints in the beginning of the solution process, as in the experimental section of [14].

[13]The instances `rnd-100` and `rnd-300` respectively correspond to the instances `random-100-bnd3` and `random-300-bnd3` from Table 4 of [14] and the stabilized Benders `Cutting-Planes` reported in [14, Table 4] the following numbers of iterations for the seven `rnd-100` instances: 235, 224, 320, 328, 408, 529, 563. For the two `rnd-300` instances, it reported 537 and 545 iterations. Compared to this enhanced `Cutting-Planes`, `Projective Cutting-Planes` still needs between $\frac{1}{4}$ and $\frac{1}{2}$ fewer iterations.

TABLE 1

Statistical comparison between `Projective Cutting-Planes` *and Benders* `Cutting-Planes`*. The first group of* (21) *rows concerns the linear relaxation of the Benders reformulation and the second group of* (10) *rows concerns the original integer Benders model.*

Instance						Projective Cutting-Planes								Standard Cutting-Planes						
Graph class	ID	$\|E\|$	$\|V\|$	$\texttt{dem}_{\max}$	OPT	Best IP Sol	Iterations			Time total [secs]			Time solve master	Iterations			Time total [secs]			Time solve master
							avg	(std)	min	avg	(std)	min		avg	(std)	min	avg	(std)	min	
`a`	1	100	20	10	42.333	48	22.8	(1)	22	0.06	(0.002)	0.06	4.4%	35	(4.9)	31	0.09	(0.01)	0.07	5.5%
`b`	1	105	60	10	245.67	265	73.8	(2.7)	72	0.2	(0.006)	0.2	6.1%	131	(11.8)	115	0.4	(0.04)	0.3	8.7%
`c`	1	110	50	10	204.33	220	56.5	(1.5)	56	0.2	(0.004)	0.2	4.9%	78.5	(16)	68	0.2	(0.05)	0.2	5.8%
`d`	1	120	60	10	299.33	317	67.5	(3)	63	0.2	(0.01)	0.2	4.3%	104	(4.3)	101	0.4	(0.02)	0.4	6.1%
`e`	1	120	30	10	67.333	77	35.4	(0.8)	35	0.1	(0.006)	0.1	4.2%	39.5	(5.5)	34	0.1	(0.02)	0.1	5.5%
`f`	1	600	90	10	281.33	306	150	(35.3)	121	6.2	(1.6)	4.8	2.5%	251	(28.8)	199	8.1	(1.1)	6.3	3.5%
`g`	1	1000	100	10	284.33	312	147	(52.1)	107	14.3	(6.1)	9.7	1.3%	310	(19.6)	278	23.4	(1.9)	20.2	3.3%
`rnd-10`	1	70	25	10	81.667	89	22.3	(0.5)	22	0.04	(<0.001)	0.04	5.3%	24	(0)	24	0.04	(<0.001)	0.04	6.2%
	2	70	25	10	64.667	74	19	(0)	19	0.04	(<0.001)	0.04	4.8%	25	(0)	25	0.04	(<0.001)	0.04	5.9%
	1	80	30	10	94	102	32.2	(1.5)	31	0.07	(0.002)	0.07	5.6%	35.2	(1.5)	34	0.07	(0.002)	0.07	6.9%
	2	80	30	10	82	91	29.5	(1.5)	29	0.06	(0.003)	0.06	5.1%	37.6	(1.2)	37	0.07	(0.003)	0.07	6.3%
	1	90	35	10	124.67	137	43.4	(1.2)	43	0.1	(0.003)	0.1	5.8%	78.5	(7.5)	71	0.2	(0.02)	0.2	7.4%
`rnd-100`	1	600	90	100	2918.7	2946	153	(20.7)	129	6.2	(1)	5	2.2%	278	(27.7)	242	9.3	(0.9)	7.9	4.5%
	2	600	90	100	2782	2810	141	(37.4)	113	5.5	(1.7)	4.3	2.4%	229	(17.9)	208	7.7	(0.7)	6.9	3.6%
	1	1000	110	100	3414.7	3452	229	(70.2)	122	23.5	(8.1)	11.5	2%	360	(23)	306	28.8	(2.3)	23.7	4.2%
	2	1000	110	100	3080.3	3112	196	(72.8)	136	19.3	(8.2)	12.6	2.3%	389	(38.2)	313	30.6	(3.7)	22.2	3.6%
	1	1500	130	100	3922	3956	382	(182)	193	89.6	(45.9)	41.1	3%	466	(20.8)	430	74.8	(4.7)	65.9	2.9%
	2	1500	130	100	4033.7	4073	282	(77.4)	190	65	(19.8)	40.1	2.2%	536	(42.5)	446	90.4	(7.8)	74.4	4.2%
	1	2000	150	100	4638	4684	259	(106)	166	103	(50.7)	59.1	1%	744	(65.8)	638	218	(18.8)	193	5.4%
`rnd-300`	1	2000	150	300	13314	13365	302	(187)	164	122	(92.6)	58.1	1.8%	744	(87.9)	626	215	(29.1)	175	4.9%
	2	2000	150	300	13358	13404	295	(172)	168	115	(76.6)	57.5	1.9%	688	(74.8)	634	202	(14.7)	184	4.6%
`a`	1	100	20	10	46		174	(27.4)	141	7.4	(5.8)	3.3	89.5%	229	(44.6)	146	9.6	(3)	4.6	95%
`b`	1	105	60	10	260		824	(206)	464	1073	(636)	379	99.5%	2987	(375)	2427	4129	(819)	2971	99.8%
`c`	1	110	50	10	214		242^{-1}	(27.1)	201	99	(31.6)	40.2	98.4%	526	(58.6)	442	378	(70.8)	284	99.6%
`d`	1	120	60	10	313		336	(53.4)	251	321	(103)	156	99.2%	1315	(184)	1049	2367	(469)	1837	99.8%
`e`	1	120	30	10	74		1336	(138)	1020	4907	(1640)	2086	99.8%	2250	(450)	1292	6703	(2857)	2450	99.9%
`rnd-10`	1	70	25	10	87		102	(11.6)	81	11.7	(5.7)	2.4	97.2%	138	(15.7)	111	12.7	(6.9)	3.2	98.4%
	2	70	25	10	72		154	(31)	121	24	(16.8)	5.9	98%	182	(30.5)	142	38.3	(19.3)	17	99.3%
	1	80	30	10	100		188	(37.8)	126	65.7	(28)	15.6	98.9%	368	(87.9)	236	183	(87.8)	45.2	99.6%
	2	80	30	10	88		214	(43.6)	105	96.5	(33.1)	60.4	99.1%	369	(58.4)	317	156	(50.4)	93.5	99.5%
	1	90	35	10	134		722^{-5}	(65.9)	664	790	(83)	629	99.5%	1962^{-2}	(188)	1748	2314	(365)	1803	99.8%

runs out of 10 fail. The running time is not perfectly proportional to the number of iterations because the structure of the ILP master problems generated along the iterations can be very different from method to method or from instance to instance.

Remark 5. Such reduction factors of 3 and 4 cannot be achieved by applying enhancement techniques (or stabilization) on the Benders `Cutting-Planes`. The last rows of Table 2 from [14] show that such enhancement techniques could lead to, respectively, $116, 165, 276$, 244, and 1128 iterations for the last five rows of Table 1 (corresponding to the first 5 instances of `random-10-bnd3` from [14, Table 2]). Compared to the number of iterations of the standard `Cutting-Planes` from Table 1, the enhancement techniques reduced the number of iterations by a factor between 1.2 and 1.75.

It is also worth noting that certain enhancement techniques could also be applied to the `Projective Cutting-Planes`; e.g., the smoothing technique [14, section 2.4.2] that consists of solving the subproblem on a smoothed query point instead of the current optimal solution $\texttt{opt}(\mathscr{P}_i)$ could apply in exactly the same manner to `Projective Cutting-Planes`. In fact, we compared above a nonstabilized `Projective Cutting-Planes` against a stabilized `Cutting-Planes`.

Finally, columns *Time solve master* contain many entries above 99%, confirming section 3.1.2: the computational bottleneck comes from the master ILPs and not from the projection or the separation subproblems that both reduce to pure LPs.

4.2. Graph coloring. We use 15 well-known instances generated during the second DIMACS implementation challenge in the 1990s [9, 12].

4.2.1. The standard coloring model. We here use the projection algorithm for standard graph coloring from section 3.2.3, based on the DLP from (3.13.a)–(3.13.e). We solve both the separation ILP (the maximum weight clique problem) and the projection DLP with similar `Branch and Bound` techniques as described in Remark 2; however, by using `cplex` in practice, the separation algorithm is significantly more sophisticated (Remark 3).

Figure 3 depicts the progress over the iterations of the lower bounds reported by `Projective Cutting-Planes` compared to those of the standard `Column Generation` on four instances.

As hinted at in section 3.2.2, both the new and the standard methods benefit from (warm-)starting the solution process using a heuristic coloring.[14] For both methods, this enables us to use from the beginning a set of initial constraints $\mathcal{A}_0 \subsetneq \mathcal{A}$ of the form $\mathbf{a}^\top \mathbf{x} \leq 1$, where $\mathbf{a} \in \mathbb{Z}^n_+$ is the incidence vector of a stable from the heuristic coloring. The heuristic coloring also enables us to construct an initial exterior point $\mathbf{d}_1$ as described in section 3.2.2. At the first iteration, `Projective Cutting-Planes` solves $\texttt{project}(\mathbf{0}_n \to \mathbf{d}_1)$, while `Column Generation` solves the separation subproblem on the same $\mathbf{d}_1$; we obtain similar (warm-)starting conditions for both methods.

We will show that both methods start from the same lower bound. By instantiating (3.12.a)–(3.12.c), the first projection subproblem $\texttt{project}(\mathbf{0}_n \to \mathbf{d}_1)$ reduces to $t^*_1 = \min\left\{\frac{1}{\mathbf{d}_1^\top \mathbf{a}} :\ \mathbf{a} \in \mathcal{P}_{0-1},\ \mathbf{d}_1^\top \mathbf{a} > 0\right\}$, where we recall that $\mathcal{P}_{0-1}$ is the set of standard stables from (3.12.c). The first pierce point is $t^*_1 \cdot \mathbf{d}_1$, and the associated first

[14]We simply used legal colorings determined in our previous work on graph coloring heuristics, available online at http://cedric.cnam.fr/~porumbed/graphs/evodiv/ or http://cedric.cnam.fr/~porumbed/graphs/tsdivint/. The associated upper bound value is provided in column 4 of Table 3.

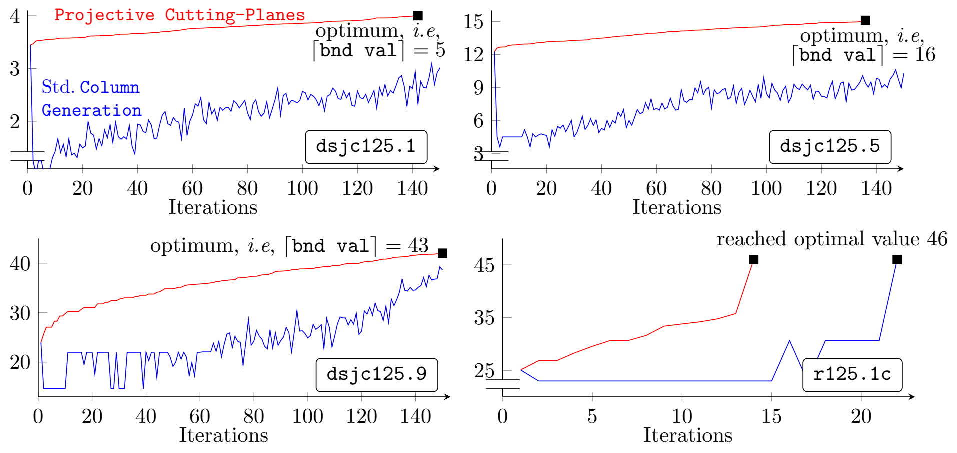

FIG. 3. *The lower bounds generated along the iterations by* `Projective Cutting-Planes` *(in red) and, respectively,* `Column Generation` *(in blue) on four instances. (Color available online.)*

lower bound is $\mathbf{b}^\top(t_1^* \cdot \mathbf{d}_1) = t_1^* \cdot \mathbf{1}_n^\top \mathbf{d}_1$, equivalent to

$$\frac{\mathbf{1}_n^\top \mathbf{d}_1}{\max\left\{\mathbf{d}_1^\top \mathbf{a} : \ \mathbf{a} \in \mathcal{P}_{0-1}\right\}}. \tag{4.1}$$

The separation subproblem in `Column Generation` is $\min\{1 - \mathbf{x}^\top \mathbf{a} : \ \mathbf{a} \in \mathcal{P}_{0-1}\}$, where $\mathbf{x}$ is the current optimal outer solution $\mathtt{opt}(\mathscr{P}_{\mathtt{it}})$ at iteration `it`. A well-known Lagrangean bound for problems with $c_a = 1 \ \forall(\mathbf{a}, c_a) \in \mathcal{A}$ is the Farley bound $\frac{\mathbf{b}^\top \mathbf{x}}{1 - m_{rdc}(\mathbf{x})}$, where $m_{rdc}(\mathbf{x})$ is the minimum reduced cost $m_{rdc}(\mathbf{x}) = \min_{(\mathbf{a},1)\in\mathcal{A}} 1 - \mathbf{x}^\top \mathbf{a}$, as described in [1, section 2.2], [17, section 3.2], [11, section 2.1], and [13, App. C]. Replacing $m_{rdc}(\mathbf{x}) = \min_{(\mathbf{a},1)\in\mathcal{A}} 1 - \mathbf{x}^\top \mathbf{a} = \min\{1 - \mathbf{x}^\top \mathbf{a} : \ \mathbf{a} \in \mathcal{P}_{0-1}\}$ for $\mathbf{x} = \mathbf{d}_1$, we obtain (4.1).

Based on the above theoretical arguments, both methods start from the same lower bound (4.1) in Figure 3 at iteration $\mathtt{it} = 1$. However, the lower bounds of `Projective Cutting-Planes` increase monotonically, while those of the standard `Column Generation` method exhibit the "yo-yo" effect. This (infamous) effect is due to the strong oscillations of the optimal solutions $\mathtt{opt}(\mathscr{P}_{\mathtt{it}})$ along the `Column Generation` iterations `it`. By stabilizing `Column Generation`, one can reduce such effects, but we are not aware of any other work in which the "yo-yo" effect could be completely eliminated. `Projective Cutting-Planes` eliminated it because each new interior solution $\mathbf{x}_{\mathtt{it}} = \mathbf{x}_{\mathtt{it}-1} + t^*_{\mathtt{it}-1}\mathbf{d}_{\mathtt{it}-1}$ is better than the previous one $\mathbf{x}_{\mathtt{it}-1}$, i.e., we have $\mathbf{b}^\top \mathbf{x}_{\mathtt{it}} > \mathbf{b}^\top \mathbf{x}_{\mathtt{it}-1}$. The objective value cannot decrease by advancing along $\mathbf{x}_{\mathtt{it}-1} \to \mathbf{d}_{\mathtt{it}-1}$ (i.e., from $\mathbf{x}_{\mathtt{it}-1}$ to $\mathbf{x}_{\mathtt{it}}$), as also stated in Step 2 in section 2.

Table 2 reports three lower bounds determined by the classical `Column Generation` (columns 2–4), followed by three bounds of `Projective Cutting-Planes` (last three columns). For both methods, the three bounds respectively correspond to the beginning (columns 2 and 6), to a midpoint (columns 3 and 7), and to the to end (columns 4 and 8) of the solution process. In fact, we tried to make the following pairs of columns report the same rounded-up bound value: columns 2 and 6; 3 and 7; 4 and 8. This explains why column 2 might actually report more than 100 iterations, i.e., `Column Generation` might need hundreds of iterations to reach the bound value obtained by `Projective Cutting-Planes` in a dozen iterations (in column 6). For each bound,

TABLE 2

The `Projective Cutting-Planes` *compared to the classical* `Column Generation` *on all standard graph coloring instances that could be solved by either method in less than* 10,000 *seconds. Both the projection and the separation subproblems are modeled and solved by* `cplex`*; regarding the projection DLP, the disjunctive constraints are implemented as logical constraints.*

Instance	Classical `Column Generation` beginning	mid iter	last iter	clique cut sz.	`Projective Cutting-Planes` beginning	mid iter	last iter
	`iter:lb/tm`	`iter:lb/tm`	`iter:lb/tm`	k	`iter:lb/tm`	`iter:ca lb/tm`	`iter:lb/tm`
`dsjc125.1`	283:3.44/29.2	380:3.8/53.8	544:4.01/135.6	4	3:3.52/33.0	78:3.80/1225	142:4.01/2809
`dsjc125.5`	253:13.08/365	306:14.27/634	378:15.08/1101	—	16:13.04/213	62:14.001/1288	136:15.003/4077
`dsjc125.9`	70:25.67/24.4	134:34.13/78	171:42.11/136	—	2:25.67/7.3	44:34.11/109	150:42.03/486
`dsjc250.9`	254:51.6/2221	275:54.98/2702	437:70.09/7757	—	5:51.06/487	34:54.01/3013	67:70.01/50185
`r125.1`	34:2.35/0.06	36:2.62/0.06	47:5/0.08	4*	5:2.35/0.18	17:2.61/0.68	20:5/0.80
`r125.1c`	15:25.09/8.18	17:30.06/8.45	22:46/9.8	—	2:26.83/4.7	6:30.06/10.4	14:46/21.6
`r125.5`	82:21.39/7.3	100:24.59/9.2	121:36/11.2	4*	3:21.21/4.2	67:24.01/74.3	116:36/136.7

* For these graphs, we added the cuts (c) from Definition 3.1. We also protected the algorithm from generating zero step lengths and stagnating: if the projection subproblem returns $t^*_{\texttt{it}} < 10^{-6}$ at some iteration `it`, we switch to a new formula to determine future inner solutions: $\mathbf{x}_{\texttt{it}} = 0.99\,(\mathbf{x}_{\texttt{it}-1} + t\,*_{\texttt{it}-1}\,\mathbf{d}_{\texttt{it}-1})$. We thus avoid the degeneracy-like issues from section 3.2.5.

we indicate the number of iterations `iter` needed to reach it, the bound value `lb`, and the CPU time `tm` in seconds. A digit in column 5 indicates that we used k-clique inequalities to accelerate the projection subproblem algorithm (see Remark 3). We excluded graphs like `le450_25c`, `le450_25d`, `le450_15c`, `le450_15d`, `dsjc500.1` simply because they require more than 10000 seconds for both methods, at least when the subproblem is solved using `cplex` (see also Remark 6, page 1028).

The first conclusion drawn from Table 2 is that, for half of the instances, `Column Generation` might need hundreds of iterations to reach the lower bounds generated by `Projective Cutting-Planes` in less than 20 iterations (compare columns 2 and 6 labeled *beginning*). This is mainly due to the monotonically increasing lower bounds of `Projective Cutting-Planes`.

Regarding the complete convergence, `Projective Cutting-Planes` is systematically faster in terms of iterations, up to reducing the number of iterations to half (`dsjc125.5`, `r125.1`) or to less than a third (on `dsjc125.1`). However, it can be slower in terms of absolute CPU times. As discussed in Remark 3, this is mostly due to the speed of the subproblem solvers provided by the `cplex` software package: the solver for the (separation) ILP is faster than the one for the (projection) DLP. By changing these solvers, the total running time can easily change.[15] Based on the arguments from Remark 2, we see no in-depth reason why the DLP needed for the projection algorithm should always be fundamentally harder in absolute terms than a similar-size ILP needed for the separation.

4.2.2. The Projective Cutting-Planes on the model with RR-stables. We here focus on the coloring model with RR-stables from section 3.2.4. We recall

[15]Indeed, although the subproblem algorithm can be seen as a black-box component in the overall design, the CPU performance of all discussed algorithms depends substantially on the running time of this subproblem algorithm. Only considering the `Column Generation`, the total running time can completely change if we replace the current `cplex` pricing (Table 2) by a pricing powered by a `Branch & Bound with Bounded Size` (BBBS) algorithm (Table 3 in section 4.2.2). On the low-density graph `dsjc125.1`, the `Column Generation` with `cplex` pricing needs 135 seconds, while the BBBS version needs 5431 seconds. The situation is inverted on a high-density graph like `dsjc125.9`: the `cplex` version needed 136 seconds and the BBBS version needed 1.61 seconds. Similar phenomena arise for the `Projective Cutting-Planes`; for instance, we could double the running time for `dsjc125.1` by simply changing the implementation of $\overline{a}_i \in \{0, \overline{\alpha}\}$ from "$\overline{a}_i \leq 0$ `or` $\overline{a}_i \geq \overline{\alpha}$" into "$\overline{a}_i = 0$ `or` $\overline{a}_i = \overline{\alpha}$." Both these equivalent constraints are implemented as logical "`or`" constraints in `cplex`.

the main ideas. The constraints $\mathcal{A}$ (the primal columns) are associated to the extreme solutions of polytope $\mathcal{P}$ from Definition 3.1 that contains all standard stables. The projection subproblem reduces now to a pure LP (3.16a)–(3.16d) that is solved by `cut generation` as indicated in section 3.2.4. We will consider two values of the parameter k which controls the size of the k-cliques used to generate reinforcing cuts (f) that define $\mathcal{P}$ in (3.14). Confirming theoretical arguments from section 3.2.4, a higher k leads to stronger lower bounds at the expense of a lower speed.

Aiming at an unbiased comparison, we prefer to solve the projection and the separation subproblems with similar techniques. In section 4.2.1, both subproblems were solved with the mathematical programming tools (based on `Branch and Bound` and continuous relaxations) of `cplex`. In the current section, we determine the maximum weight stables for the separation subproblem using the same `Branch & Bound with Bounded Size` (BBBS) algorithm used by the `Projective Cutting-Planes` to find k-cliques when constructing $\mathcal{P}$ in (3.14). For the standard `Column Generation`, the maximum stable of the considered graph (column 5 in Table 3) is given as input to BBBS, i.e., it represents the bounded size given as input to BBBS.

Table 3 compares the standard and the new methods, placing an emphasis on three lower bounds reported along the iterations. The first four columns describe the instance: the density in column 1, the graph in column 2, $|V|$ in column 3, and the heuristic upper bound in column 4. Columns 6–8 report three lower bounds obtained by the standard `Column Generation` along the iterations (each table cell in these columns indicates the bound value and the required CPU time). For `Projective Cutting-Planes`, we consider in column 9 two values of the parameter k used to generate k-cliques to construct $\mathcal{P}$. Columns 10–12 provide three lower bounds of the `Projective Cutting-Planes` in the same format as in columns 6–8.

The last column of Table 3 reports the result of an additional special iteration: take the last pierce point reported by the `Projective Cutting-Planes` with RR-stables (next-to-last column), multiply it with $\alpha = 0.9999$ (for the reasons indicated in section 3.2.5), and project it towards $\mathbf{1}_n$ *in the original model* with standard stables. This last projection may lead to an even better lower bound.

Remark 6. The state-of-the-art `Column Generation` algorithm for graph coloring from [9] could not converge in less than three days for instances like `le450_25c`, `le450_25d`, `le450_15c`, `le450_15d`, and `dsjc500.1`. The difficulty of these instances is confirmed by our numerical experiments: our `Column Generation` can indeed "stall" on such instances, exactly for the reason indicated in [9], namely, "the maximum-weight stable-set problems that need to be solved exactly become too numerous and too difficult." More generally, low-density graphs like the above are often quite difficult for the standard `Column Generation` because they have very large stables that can be very hard to generate (at each call to the separation subproblem). For such graphs, our method obtained certain successes:

1. For `le450_25c`, `le450_25d`, `le450_15c`, and `le450_15d`, `Projective Cutting-Planes` reported a lower bound that matches the chromatic number in less than one hour, which seems out of reach for the standard `Column Generation`. Although these instances are not very hard in absolute terms because they can be solved with other external methods based on matching the maximum clique size with an upper bound, the lower bounds of `Projective Cutting-Planes` are more general. They could work in the same manner when the above external methods fail, e.g., for a (dual) objective function $\mathbf{b} \neq \mathbf{1}_n$, as in a multicoloring problem.[16]

[16]The chromatic number of such graphs can be more easily determined by exploiting the fact that the maximum clique size may match an upper bound found by a (meta-)heuristic. When $\mathbf{b} \neq \mathbf{1}_n$, this method fails because the maximum clique is no longer a lower bound.

TABLE 3

Comparison of three lower bounds obtained by the standard `Cutting-Planes` *(on the original coloring model) and by the* `Projective Cutting-Planes` *(on the coloring model with RR-stables) along the iterations. For the* `Projective Cutting-Planes`*, the last column reports the rounded-up bound obtained by performing a last projection in the original model with 0–1 stables: project the last pierce point multiplied by $\alpha = 0.9999$ towards $\mathbf{1}_n$; * indicates that we could only compute a lower bound on this last step length (using* `cplex`*). The lower bounds that equal the value of the given heuristic (column 4) are marked in bold; when the* `Projective Cutting-Planes` *closes the gap this way, the last column indicates "optim ended" because performing an additional iteration would become useless.*

Density	Instance	n	Upper bound	Opt max stable	Standard `Column Generation` with standard stables			`Projective Cutting-Planes` with RR stables				one last iter with std. stab.
					lb/tm	lb/tm	lb/tm	clq. sz. k	lb/tm	lb/tm	lb/tm	$\lceil$lb$\rceil$/tm
0.094	`dsjc125.1`	125	5	35	3.45/1.5	3.52/760	4.01/5431	3	2.95/0.43	3.01/2.77	4/15.13	4/21.45
								4	2.95/0.53	3.01/1.22	4/15.51	5/21.06
0.17	`le450_25c`	450	25	47	2.5/5.03	5.2/195	6.87/2535	5	9.11/2.34	10.01/41	**25**/1242	optim ended
								25	9.11/2.28	10.01/43	**25**/1412	optim ended
0.17	`le450_25d`	450	25	43	2.77/5.26	5.45/409	6.15/2440	5	10.21/3.28	13.01/224	**25**/1364	optim ended
								25	10.21/3.14	13.01/210	**25**/1393	optim ended
0.24	`p_hat300-1`	300	19	39	2.11/3.4	7.36/206	10.19/5418	3	4.54/1.58	6.01/86	8/292	9/8479
								8	6.01/33	7.01/160	8/267	9/3212
0.50	`dsjc125.5`	125	18	35	4.5/2.25	10.01/2.44	15.06/4.88	7	7.91/0.66	8.01/0.89	9/8.11	14/76
								10	8.12/2.50	8.95/13	10/13.3	11/62
0.90	`dsjc125.9`	125	44	4	24/1.54	40.03/1.60	42.26/1.61	10	16.25/2.68	24/52	25/259	26/265
								∞	24/7.75	29.06/747	31.01/5497	32/5812
0.10	`dsjc250.1`	250	9	70	2.01/2.28	3.05/270	3.70/9180	3	3.5/0.35	3.75/10.9	3.81/18.2	5*/ 1076
								4	3.5/0.46	3.7/6.1	3.75/9.8	5*/91
0.50	`dsjc250.5`	250	28	12	5.6/4.93	15.04/10.34	25.01/225	6	8.24/7.03	9.01/13.36	10.001/119	21/5921
								∞	9.98/83	10.01/86	12/391	13/2492
0.90	`dsjc250.9`	250	73	5	34.3/10.4	60.01/11.18	70.09/11.6	10	10/3.17	10/3.17	10/3.17	50/586
								∞		time out (>10000)		–
0.10	`dsjc500.1`	500	12	122	2.25/5.14	2.50/66	3.06/3376	3	3.55/4.19	4.01/272	5/340	6*/5514
								5	4.08/79	4.50/295	5/682	6*/5857
0.03	`r125.1`	125	5	49	2.31/0.02	**5**/0.2	**5**/0.2	2	2.31/0.23	2.51/5.67	**5**/17.7	optim ended
								5	2.31/0.26	2.55/2.61	**5**/10.6	optim ended
0.97	`r125.1c`	125	46	7	23/2.66	34.5/2.67	**46**/2.67	10	10.55/0.42	11.27/0.75	16/2.56	17/6.1
								∞	25.01/1.85	34.2/13.26	**46**/28.64	optim ended
0.50	`r125.5`	125	36	5	20.57/0.09	25.2/0.22	**36**/0.25	30	20.57/0.74	25.04/7.80	30/59.45	30/157.8
								36	20.57/0.74	25.01/7.84	**36**/15.57	optim ended
0.17	`le450_15c`	450	15	49	3.07/6.23	4.22/145	5.32/1172	3	8.34/6.74	10.01/308	**15**/1751	optim ended
								15	9.19/14.52	13.01/210	**15**/3331	optim ended
0.17	`le450_15d`	450	15	49	3.07/6.33	4.23/145	5.33/1172	3	8.21/5.53	10.01/359	**15**/2326	optim ended
								15	9.04/11.6	12.01/2425	**15**/3786	optim ended

2. For `dsjc500.1`, the last projection in the model with standard stables (last column of Table 3) reports a (rounded-up) lower bound of 6; to the best of our knowledge [12, 9], this is the first time a feasible solution of such quality could be found using a `Column Generation` model. As in the case of the four graphs in point 1 above, the bound value in itself has been already discovered, but only using external methods (based on constructing a reduced induced subgraph in [9]). We can even describe this feasible solution of (3.11): assign $0.9999 + 0.00000101$ to the vertices 4, 30, 47, 361, 475 and 0.00000101 to all remaining 495 vertices.[17]
3. For `dsjc250.1`, the last column of Table 3 indicates that the (`cplex` solver for the) last projection showed that the last step length t^*_{last} is large enough to prove $\lceil 3.80585222 + t^*_{\text{last}} \cdot 250 \rceil = 6$, where 3.80585222 is the value of the last pierce point multiplied by $\alpha = 0.9999$. By allowing more time, `cplex` reported (after about 36 hours using up to 20 threads on a multicore CPU) a lower bound of 0.0088 on the last step-length, i.e., it proved $t^*_{\text{last}} > 0.0088$. The last projection with standard stables thus proves a lower bound of $\lceil 3.80585222 + t^*_{\text{last}} \cdot 250 \rceil \geq \lceil 3.80585222 + 0.0088 \cdot 250 \rceil \geq \lceil 6.005 \rceil = 7$. We see no risk of numerical errors because after 56 hours, `cplex` even proved $t^*_{\text{last}} > 0.01$. To the best of our knowledge [12, 9], this is the first time a lower bound of 7 has ever been reported on this graph.

5. Conclusion and prospects.

We proposed a new method to optimize LPs over polytopes $\mathscr{P}$ with unmanageably many constraints. The key idea is to "upgrade" the separation subproblem used in `Cutting-Planes` to a more general projection subproblem. Given an arbitrary inner (feasible) solution $\mathbf{x} \in \mathscr{P}$ and a direction $\mathbf{d} \in \mathbb{R}^n$, this subproblem asks to determine the pierce (first-hit) point $\mathbf{x} + t^*\mathbf{d}$ encountered when advancing from $\mathbf{x}$ along $\mathbf{d}$. The proposed `Projective Cutting-Planes` generates a sequence of inner solutions $\mathbf{x}_{\texttt{it}}$ and a sequence of outer solutions $\texttt{opt}(\mathscr{P}_{\texttt{it}})$ that both converge to an optimal solution $\texttt{opt}(\mathscr{P})$ along the iterations `it`. `Projective Cutting-Planes` can offer *several advantages*.

– The convergent sequence of feasible solutions $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots \in \mathscr{P}$ is generated using a *built-in* mechanism (the projection algorithm). There is no such built-in functionality in `Cutting-Planes`: even if one can sometimes use various ad hoc methods to calculate feasible solutions in a standard `Cutting-Planes` (e.g., the Farley bound in `Column Generation`), these inner solutions usually remain a by-product of the algorithm, and they do *not* usually "drive" the `Cutting-Planes` evolution. In `Projective Cutting-Planes`, the inner solutions $\mathbf{x}_{\texttt{it}}$ generated along the iterations `it` are a vital component that do guide the algorithm evolution.

– The numerical experiments showed a significant potential to reduce the computing effort needed to fully converge. In section 4.1, both the number of iterations and the CPU time could be reduced by a factor of up to 3 or 4 (e.g., for instances `b` and `d` in the last rows of Table 1). In graph coloring (section 4.2.1), the reduction of the number of iterations can also reach a factor of 4 (first instance in Table 2). We could even find lower bounds that have never been reported before on the (well-studied) graph coloring problem (see points 2 and 3 of Remark 6, page 1028). Further numerical tests on *Multiple-Length Cutting-Stock* in [15, Table 2] show a reduction of a factor of 2 on a few difficult instances, which seems beyond the potential of

[17]The associated objective value is $5 \cdot 0.9999 + 500 \cdot 0.00000101 = 4.9995 + 0.000505 = 5.0005$. Without any risk of numerical errors, we can prove this solution is feasible because there is no stable of size 100 that contains any of the vertices 4, 30, 47, 361, or 475 (these vertices form a clique). Indeed, `cplex` showed in less than two hours that the size of such a stable is upper bounded by 99; the above solution is thus feasible because $0.9999 + 99 \cdot 0.00000101 = 0.99999999 < 1$.

more classical stabilization methods.[18]

– By defining $\mathbf{x}_{\mathtt{it}}$ as the best solution ever found (the last pierce point), one can prove that the lower bounds $\mathbf{b}^\top \mathbf{x}_{\mathtt{it}}$ become strictly increasing along the iterations `it`. This way, the lower bounds for graph coloring (Figure 3) eliminated the infamous "yo-yo" effect arising in most (if not all) existing `Column Generation` algorithms.

There are also certain (inherent) deterrents to adopting the new method. First, it can be more difficult to design a projection algorithm than a separation one, because the projection subproblem is more general. As such, more work may be needed to make `Projective Cutting-Planes` reach its full potential. Second, we do not (yet) have a fully comprehensive insight into why `Projective Cutting-Planes` is more successful on some problems than on others. It remains rather difficult to explain why $\alpha < 0.5$ is often better than $\alpha = 1$ when choosing the inner solution $\mathbf{x}_{\mathtt{it}}$ via $\mathbf{x}_{\mathtt{it}} = \mathbf{x}_{\mathtt{it}-1} + \alpha \cdot t^*_{\mathtt{it}-1}\mathbf{d}_{\mathtt{it}-1}$. Still, we can advance the following arguments:

– In a successful `Projective Cutting-Planes` implementation, the feasible solutions $\mathbf{x}_{\mathtt{it}}$ generated along the iterations `it` are rather well centered, i.e., they do *not* exhibit a "bang-bang" behavior with strong oscillations. In a loose sense, the inner solutions $\mathbf{x}_{\mathtt{it}}$ are reminiscent of an interior point algorithm in which the solutions follow a central path [8, section 3.3]. The outer solutions $\mathtt{opt}(\mathscr{P}_{\mathtt{it}})$ are reminiscent of the Simplex algorithm that is known to exhibit a "bang-bang" behavior when moving along edges from one extreme solution to another. We can argue that, by choosing $\mathbf{x}_{\mathtt{it}} = \mathbf{x}_{\mathtt{it}-1} + \alpha \cdot t^*_{\mathtt{it}-1}\mathbf{d}_{\mathtt{it}-1}$ with $\alpha < 0.5$, `Projective Cutting-Planes` generates more well-centered interior paths, limiting the "bang-bang" effects. This idea is further explored with numerical tests in [15, section 3.3].

– The projection subproblem may generate stronger constraints than the separation subproblem. As described in section 2.4.1 of [14], when $\mathbf{x} = \mathbf{0}_n$, the projection subproblem $\mathtt{project}(\mathbf{x} \to \mathbf{d})$ is equivalent to normalizing all constraints (to make them all have the same right-hand side value) and then choosing one by separating $\mathbf{x} + \mathbf{d}$. Even if this paper uses $\mathbf{x} \neq \mathbf{0}_n$, the projection subproblem can still generate stronger (normalized) constraints than the separation subproblem.[19]

The proposed method could be potentially useful in solving other LPs with prohibitively many constraints, beyond the four problems addressed in this paper or in the follow-up work [15], i.e., `Projective Cutting-Planes` could be implemented whenever it is possible to design a projection algorithm whose running time is similar to that of the separation algorithm. We thus hope this work can shed useful light on solving such large-scale LPs and help one overcome certain limitations of the current practices used in standard `Cutting-Planes`.

Acknowledgment. We thank the editor and the referees for their work on this paper.

[18]Notice that the standard `Column Generation` needs two times more iterations than `Projective Cutting-Planes` on the `hard` instances from [15, Table 2]; on a quarter of instances from this table `Column Generation` needs at least 1.6 more iterations than `Projective Cutting-Planes`. The potential of stabilization methods seems limited to a factor of 1.2 except for the easier instances `m20` and `m35`, at least based on the experiments from Table 2 of [16].

[19]Consider choosing between $2x_1+3x_2 \leq 1$ and $200x_1+300x_2 \leq 495$. When solving the separation subproblem on $[1\ 1]^\top$, the second constraint might seem more violated because $200 + 300 - 495 = 5 > 2 + 3 - 1 = 4$. But the (level sets of the) two constraints are parallel and the second constraint is considerably weaker, even redundant. It is not difficult to check that the projection subproblem can never return this (redundant) second constraint for any feasible $\mathbf{x}$ and for any direction $\mathbf{d} \in \mathbb{R}^2$.

REFERENCES

[1] H. B. Amor and J. M. V. de Carvalho, *Cutting stock problems*, in Column Generation, G. Desaulniers, J. Desrosiers, and M. M. Solomon, eds., Springer, Boston, MA, 2005, pp. 131–161.

[2] J. F. Benders, *Partitioning procedures for solving mixed-variables programming problems*, Numer. Math., 4 (1962), pp. 238–252.

[3] A. Charnes and W. W. Cooper, *Programming with linear fractional functionals*, Naval Res. Logist. Quart., 9 (1962), pp. 181–186.

[4] F. Clautiaux, C. Alves, and J. M. V. de Carvalho, *A survey of dual-feasible and superadditive functions*, Ann. Oper. Res., 179 (2009), pp. 317–342.

[5] A. M. Costa, *A survey on Benders decomposition applied to fixed-charge network design problems*, Comput. Oper. Res., 32 (2005), pp. 1429–1450.

[6] M. Fischetti and M. Monaci, *Cutting plane versus compact formulations for uncertain (integer) linear programs*, Math. Program. Comput., 4 (2005), pp. 239–273.

[7] J. Gondzio, *Interior point methods* 25 *years later*, European J. Oper. Res., 218 (2012), pp. 587–601.

[8] J. Gondzio, P. González-Brevis, and P. Munari, *Large-scale optimization with the primal-dual column generation method*, Math. Program. Comput., 8 (2016), pp. 47–82.

[9] S. Held, W. Cook, and E. C. Sewell, *Maximum-weight stable sets and safe lower bounds for graph coloring*, Math. Program. Comput., 4 (2005), pp. 363–381.

[10] A. N. Letchford, F. Rossi, and S. Smriglio, *The stable set problem: Clique and nodal inequalities revisited*, submitted.

[11] M. E. Lübbecke and J. Desrosiers, *Selected topics in column generation*, Oper. Res., 53 (2005), pp. 1007–1023.

[12] E. Malaguti, M. Monaci, and P. Toth, *An exact approach for the vertex coloring problem*, Discrete Optim., 8 (2011), pp. 174–190.

[13] D. Porumbel, *Ray projection for optimizing polytopes with prohibitively many constraints in set-covering column generation*, Math. Program., 155 (2016), pp. 147–197.

[14] D. Porumbel, *From the separation to the intersection subproblem for optimizing polytopes with prohibitively many constraints in a Benders decomposition context*, Discrete Optim., 29 (2018), pp. 148–173.

[15] D. Porumbel, `Projective Cutting-Planes` *for Robust Linear Programming and Cutting-Stock Problems*, Technical report, CEDRIC CS Lab CEDRIC-19-4550, to be submitted for publication (2020), http://cedric.cnam.fr/~porumbed/papers/techrep4550.pdf

[16] D. Porumbel and F. Clautiaux, *Constraint aggregation in column generation models for resource-constrained covering problems*, INFORMS J. Comput., 29 (2017), pp. 170–184.

[17] F. Vanderbeck, *Computational study of a column generation algorithm for bin packing and cutting stock problems*, Math. Program., 86 (1999), pp. 565–594.