From the separation to the intersection sub-problem in Benders decomposition models with prohibitively-many constraints

Daniel Porumbel

From the Separation to the Intersection Sub-problem in Benders Decomposition Models with Prohibitively-Many Constraints

Daniel Porumbel*
*CEDRIC CS Lab, CNAM, 292 rue Saint-Martin, F-75141 Paris, France

Abstract

We consider a minimization linear program over a polytope $\mathscr{P}$ described by prohibitively-many constraints. Given a ray direction $\mathbf{0} \rightarrow \mathbf{r}$, the intersection sub-problem asks to find: (i) the intersection point $t^*\mathbf{r}$ between the ray and the boundary of $\mathscr{P}$ and (ii) a constraint of $\mathscr{P}$ satisfied with equality by $t^*\mathbf{r}$. In [12, §2], we proposed a method based on the intersection sub-problem to optimize general linear programs. In this study, we use a `Cutting-Planes` method in which we simply replace the separation sub-problem with the intersection sub-problem. Although the intersection sub-problem is more complex, it is not necessarily computationally more expensive than the separation sub-problem and it has other advantages. The main advantage is that it can allow the `Cutting Planes` algorithm to generate a feasible solution (using $t^*\mathbf{r} \in \mathscr{P}$) at each iteration, which is not possible with a standard separation sub-problem. Solving the intersection sub-problem is equivalent to normalizing all cuts and separating; this interpretation leads to showing that the intersection sub-problem can find stronger cuts. We tested such ideas in a Benders decomposition model with prohibitively-many feasibility cuts. We show that under certain (mild) assumptions, the intersection sub-problem can be solved within the same asymptotic running time as the separation one. We present numerical results on a network design problem that asks to install a least-cost set of links needed to accommodate a one-to-many flow.

1 Introduction

Let us start with a general Integer Linear Program (ILP) often arising in Benders reformulations:[1]

$$\min\left\{\mathbf{d}^\top\mathbf{y} : \mathbf{u}^\top\mathbf{y} \geq b,\ \forall(\mathbf{u},b) \in \mathcal{C},\ \mathbf{y} \in \mathbb{Z}^n_+\right\} = \min\left\{\mathbf{d}^\top\mathbf{y} : \mathbf{y} \in \mathscr{P},\ \mathbf{y} \in \mathbb{Z}^n_+\right\}, \tag{1.1}$$

where $\mathcal{C}$ is a set of rows (constraints). We do not formally impose any condition on the size of $\mathcal{C}$, but we consider that listing all rows is computationally very exhausting, if not impossible. As such, practical algorithms for this ILP only manipulate a subset of $\mathcal{C}$. A standard `Cutting-Planes` or `Branch-and-cut` optimizes the above ILP by progressively removing infeasibility. For each intermediate optimal solution $\mathbf{y}$, one solves the separation sub-problem to (try to) separate $\mathbf{y}$ and to add a new constraint of $\mathcal{C}$. A disadvantage of the canonical `Cutting-Planes` is that it reports *no* feasible solution before the end of the convergence.

The standard `Cutting-Planes` belongs to the class of *dual (outer) methods*, because it converges to the optimum of the above LP through a sequence of infeasible (outer, dual) solutions. In contrast, *a primal method* constructs a converging sequence of feasible (interior) solutions. The primal and dual methods are described in greater detail in [12, §1.1.1.1]. The approach proposed in this paper can be seen as a *primal and dual* method, because it generates a convergent sequence of both feasible and infeasible solutions.

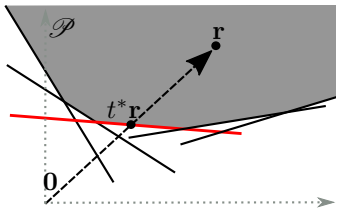


Figure 1: An intuitive view of a ray projection. The first-hit constraint is depicted in red.

Given a ray direction $\mathbf{0} \rightarrow \mathbf{r}$, the intersection sub-problem asks to find the minimum $t^* \geq 0$ such that $\mathbf{u}^\top(t^*\mathbf{r}) \geq b,\ \forall(\mathbf{u},b) \in \mathcal{C}$. If $b > 0\ \forall(\mathbf{u},b) \in \mathcal{C}$ (so that $\mathbf{0}$ is infeasible), this is a generalization of the separation sub-problem: if $t^* \leq 1$, then $\mathbf{r}$ satisfies all constraints in $\mathcal{C}$ (as in the example from Figure 1); otherwise, $\mathbf{r}$ is infeasible and can be separated by a constraint of $\mathcal{C}$. Besides

[1] The condition $\mathbf{y} \in \mathbb{Z}^n_+$ will be lifted in Section 4.3, which amounts to solving a Linear Program (LP) instead of an ILP.

the value of t^*, the intersection sub-problem asks to find a first-hit constraint $\mathbf{u}^\top \mathbf{y} \geq b$ satisfied with equality by $\mathbf{y} = t^*\mathbf{r}$. This first-hit constraint separates all points $t\mathbf{r}$ with $t < t^*$ from the feasible polytope $\mathscr{P}$.

Let us briefly compare the intersection and the separation sub-problems. A `Cutting-Planes` algorithm works with a relaxed version of (1.1) using only a subset of the constraints $\mathcal{C}$, *i.e.*, a relaxed master ILP. Given the current optimum solution $\mathbf{r}$ of this relaxed master ILP, the separation sub-problem asks to maximize:

$$\max_{(\mathbf{u},b)\in\mathcal{C}} b - \mathbf{u}^\top \mathbf{r}. \tag{1.2}$$

If the result of this sub-problem is greater than 0 for some $(\mathbf{u}_i, b_i) \in \mathcal{C}$, then $\mathbf{r}$ does not satisfy $\mathbf{u}_i^\top \mathbf{y} \geq b_i$, and so, a `Cutting-Planes` algorithm would separate $\mathbf{r}$ by adding $\mathbf{u}_i^\top \mathbf{y} \geq b_i$ to the current constraint set. Then, it would re-optimize the new relaxed master ILP using the new enlarged set of constraints.

We now give a short proof that the intersection sub-problem along $\mathbf{0} \to \mathbf{r}$ reduces to maximizing the ratio below, assuming $\mathbf{u}^\top \mathbf{r} > 0, b > 0\ \forall (\mathbf{u}, b) \in \mathcal{C}$ to avoid unnecessary complication in the introduction.[2]

$$t^* = \max_{(\mathbf{u},b)\in\mathcal{C}} \frac{b}{\mathbf{u}^\top \mathbf{r}}. \tag{1.3}$$

We associate each $(\mathbf{u}_i, b_i) \in \mathcal{C}$ to a value $t_i = \frac{b_i}{\mathbf{u}_i^\top \mathbf{r}} > 0$, so that $t_i\mathbf{r}$ satisfies with equality the constraint $\mathbf{u}_i^\top \mathbf{y} \geq b_i$. Taking $t^* = \max_{(\mathbf{u}_i,b_i)\in\mathcal{C}} t_i$, we obtain $t^* \geq t_i$ for all $(\mathbf{u}_i, b_i) \in \mathcal{C}$, and so, $\mathbf{u}_i^\top (t^*\mathbf{r}) \geq \mathbf{u}_i^\top (t_i\mathbf{r}) = b_i$, where we used $\mathbf{u}_i^\top \mathbf{r} \geq 0$. This shows that $t^*\mathbf{r}$ satisfies all constraints in $\mathcal{C}$, *i.e.*, $t^*\mathbf{r} \in \mathscr{P}$. We still need to show that t^* is minimum with this property. This follows from the fact that $t^*\mathbf{r}$ satisfies $\mathbf{u}_i^\top (t^*\mathbf{r}) = b_i$ for some $(\mathbf{u}_i, b_i) \in \mathcal{C}$ that maximizes (1.3). As such, any $t < t^*$ would lead to $\mathbf{u}_i^\top (t\mathbf{r}) < b_i$, violating the constraint $\mathbf{u}_i^\top \mathbf{y} \geq b_i$. A similar result in the context of a maximization problem can be found in Proposition 3 of [12, §3.2]. Comparing (1.2) and (1.3), the intersection sub-problem can be seen as a generalized version of the separation sub-problem. We will also discuss in Section 2.5.1 that solving the intersection sub-problem is equivalent to normalizing all constraints (*i.e.*, make them all have a right-hand side term of 1) followed by choosing one constraint by classical separation.

Since the invention of the Benders decomposition in 1962 [2], the approach has become increasingly popular in optimization and hundreds of papers have used it for a wide variety of applications.[3] In particular, the Benders reformulation has been very successful for network design problems [5, 6, 7, 8]. The prohibitively-many constraints $\mathcal{C}$ correspond to the extreme solutions (optimality cuts) and the extreme rays (feasibility cuts) of a polytope $\mathcal{P}$ referred to as the Benders sub-problem polytope. To solve the separation sub-problem, one optimizes a Linear Program (LP) over $\mathcal{P}$. We will see (Section 2.3) that the intersection sub-problem reduces to solving a linear-fractional program over $\mathcal{P}$, maximizing an objective like (1.3). This can be done by casting the linear-fractional program into an LP using the Charnes-Cooper transformation [3]. Based on this approach, the computational complexity of the intersection sub-problem algorithm is the same as that of the separation algorithm, *i.e.*, it is the complexity of solving an LP over $\mathcal{P}$.

Notice that a fractional feasible solution $t^*\mathbf{r} \in \mathscr{P}$ determined by the intersection sub-problem might *not necessarily respect the integrality* constraint $t^*\mathbf{r} \in \mathbb{Z}^n_+$ imposed by (1.1). However, the Benders decomposition models discussed in this paper use integer master variables to indicate a number of times that a transmission facility (*e.g.*, a cable) is installed and one can obtain an integer feasible solution by rounding up all components of $t^*\mathbf{r}$. There is no natural constraint in $\mathscr{P}$ that forbids an increase (by rounding) of the number of installed facilities, see Observation 4 at the end of Section 3.3 for an explicit application example.

The method proposed in this paper is specifically designed to solve large-scale ILPs (1.1) of a particular form, arising in Benders reformulations. However, the most general intersection ideas could be potentially useful for other problems that fit well the general ILP (1.1). The necessary condition is to have an intersection sub-problem algorithm and to be able to apply a rounding procedure as above. Appendix A presents two problems (using no Benders decomposition) that fit well the ILP (1.1) and that surely allow a rounding

[2] In Theorem 1 of Section 2.3 we solve the intersection sub-problem by addressing all degenerate cases.

[3] The reader whose curiosity is piqued can further relate to surveys [4, 13] and to the references therein. As early as 1981, [9] compiled a list of successful applications (see page 1) such as scheduling the movement of railway engines, airline routing, industrial distribution systems, or vehicle routing. Many more other examples can be found in more recent work [4, 13].

procedure because their constraints $\mathcal{C}$ only have *non-negative* coefficients. Other problems with positive and negative coefficients would *not* allow such a rounding. Finally, the intersection ideas are most useful when $\mathbf{0}$ is infeasible (*i.e.*, $\exists(\mathbf{u}, b) \in \mathcal{C}$ such that $b > 0$) and the feasible solutions have positive objective values.

The remaining is organized as follows. Section 2 presents the main theoretical description from this paper: the study of the intersection sub-problem in a Benders reformulation model with feasibility cuts; this includes algorithmic aspects of the Benders' cut generating ILP, *e.g.*, the use of solution smoothing techniques to accelerate the convergence (both for separation and intersection sub-problems). Section 3 is devoted to a network design application example and its specific intersection sub-problem. Section 4 provides numerical results on this network design problem, followed by conclusions in the last section. Appendix A presents two examples of other (non Benders) ILP models for problems that fit well the general ILP (1.1). Appendix B describes a generalized intersection sub-problem algorithm, for a Benders decomposition model with both optimality and feasibility cuts, with no restriction on the projected or flow costs.

2 The intersection sub-problem in a Benders decomposition context and advanced `Cutting-Planes`

2.1 The classical Benders decomposition

In order to (try to) keep the paper self-contained, let us briefly introduce the main steps of the general Benders decomposition approach [2, 4]. We consider integer variables $\mathbf{y}$ and fractional variables $\mathbf{x}$. As a general example, in network design (resp. facility location problems) $\mathbf{y}$ might encode the placement of transmission (resp. production) facilities and $\mathbf{x}$ could quantify flows (resp. delivered goods or products). The general Mixed Integer Linear Program (MILP) is:

$$\min \mathbf{d}^\top \mathbf{y} + \mathbf{c}^\top \mathbf{x} \tag{2.1a}$$
$$\mathbf{D}\mathbf{y} \geq \mathbf{e} \tag{2.1b}$$
$$\mathbf{B}\mathbf{y} + \mathbf{A}\mathbf{x} \geq \mathbf{b} \tag{2.1c}$$
$$\mathbf{y} \in \mathbb{Z}^n_+,\ \mathbf{x} \geq \mathbf{0} \tag{2.1d}$$

where some of the constraints (2.1c) can act on $\mathbf{x}$ only, *i.e.*, $\mathbf{B}$ can have some null rows.

We now present the main steps of the Benders decomposition, essentially following the reasoning from [4, §2]. Based on the below reformulation of the above program, we will dualize the inner LP of (2.2c).

$$\min \mathbf{d}^\top \mathbf{y} + \hat{z} \tag{2.2a}$$
$$\mathbf{D}\mathbf{y} \geq \mathbf{e} \tag{2.2b}$$
$$\hat{z} = \min\left\{\mathbf{c}^\top \mathbf{x} : \mathbf{B}\mathbf{y} + \mathbf{A}\mathbf{x} \geq \mathbf{b},\ \mathbf{x} \geq \mathbf{0}\right\} \tag{2.2c}$$
$$\mathbf{y} \in \mathbb{Z}^n_+ \tag{2.2d}$$

Introducing dual variables $\mathbf{u}$ in the inner LP (considered with $\mathbf{x}$ as decision variables and $\mathbf{y}$ as parameters), the primal-dual linear programming properties lead to

$$\hat{z} = \max\{(\mathbf{b} - \mathbf{B}\mathbf{y})^\top \mathbf{u} :\ \mathbf{u} \in \mathcal{P}\}, \tag{2.3}$$

where

$$\mathcal{P} = \left\{\mathbf{u} \geq \mathbf{0} : \mathbf{A}^\top \mathbf{u} \leq \mathbf{c}\right\} \tag{2.4}$$

is the *Benders sub-problem polytope* that does not depend on the current $\mathbf{y}$. The optimal $\hat{z}$ in (2.3) can be modelled as a decision variable that is bounded from below by $\hat{z} \geq (\mathbf{b} - \mathbf{B}\mathbf{y})^\top \mathbf{u}\ \forall \mathbf{u} \in \mathcal{P}$ and can be interpreted as a projected cost (associated to $\mathbf{y}$) to be minimized. We suppose $\mathcal{P}$ is always not empty (*e.g.*, $\mathcal{P}$ surely contains $\mathbf{u} = \mathbf{0}$ if $\mathbf{c} \geq \mathbf{0}$), because otherwise the primal (2.2c) would be infeasible or unbounded. As such, $\mathcal{P}$ can be described by its (prohibitively-many) vertices and extreme rays. All extreme rays $\mathbf{u}^e \in \mathcal{P}$

satisfy $0 \geq (\mathbf{b} - \mathbf{B}\mathbf{y})^\top \mathbf{u}^e$ for a feasible $\mathbf{y}$, because otherwise $\hat{z}$ would be unbounded. The above MILP (2.2a)-(2.2d) can thus be written in the following *Benders decomposition* form:

$$\min \mathbf{d}^\top \mathbf{y} + \hat{z} \tag{2.5a}$$
$$\mathbf{D}\mathbf{y} \geq \mathbf{e}$$
$$\hat{z} \geq (\mathbf{b} - \mathbf{B}\mathbf{y})^\top \mathbf{u}^i, \quad \text{for any vertex } \mathbf{u}^i \in \mathcal{P} \tag{2.5b}$$
$$0 \geq (\mathbf{b} - \mathbf{B}\mathbf{y})^\top \mathbf{u}^e, \quad \text{for any extreme ray } \mathbf{u}^e \in \mathcal{P} \tag{2.5c}$$
$$\mathbf{y} \in \mathbb{Z}^n_+, \hat{z} \in \mathbb{R} \tag{2.5d}$$

Observation 1. *To solve the above program* (2.5a)-(2.5d)*, the* `Benders Cutting-Planes` *algorithm delays the generation of the constraints* (2.5b)-(2.5c) *and works at each iteration with a relaxed master MILP containing a reduced constraint set. Given the current optimal solution* $(\mathbf{y}, \hat{z})$ *of the relaxed master MILP at the current iteration, the* `Cutting-Planes` *algorithm executes the following steps:*

(A) solve the LP $\max\{(\mathbf{b} - \mathbf{B}\mathbf{y})^\top \mathbf{u} : \ \mathbf{u} \in \mathcal{P}\}$ *from (2.3), in an attempt to separate the current optimal solution* $(\mathbf{y}, \hat{z})$*. If* $\hat{z} < \max\{(\mathbf{b} - \mathbf{B}\mathbf{y})^\top \mathbf{u} : \ \mathbf{u} \in \mathcal{P}\}$*, one has to insert a constraint* (2.5b) *or* (2.5c) *associated to the optimal* $\mathbf{u} \in \mathcal{P}$ *to separate* $(\mathbf{y}, \hat{z})$*; see also below for more details on the case of extreme rays.*

(B) re-optimize the new relaxed master MILP (enriched with the above-generated constraint) to find a new optimal solution $(\mathbf{y}, \hat{z})$*.*

(C) repeat from Step (A), until the optimal solution $(\mathbf{y}, \hat{z})$ *can no longer be separated, i.e., until* $(\mathbf{y}, \hat{z})$ *becomes optimal in* (2.5a)-(2.5d)*.*

Notice that Step (A) solves the separation sub-problem on $(\mathbf{y}, \hat{z})$. In the worst case, it can find an extreme ray $\mathbf{u}^e$ that does not respect the feasibility cut (2.5c), making $\max\{(\mathbf{b} - \mathbf{B}\mathbf{y})^\top \mathbf{u} : \ \mathbf{u} \in \mathcal{P}\}$ reach an arbitrarily large value, *i.e.*, $\hat{z}$ becomes unbounded and the current $\mathbf{y}$ can be considered infeasible. In such a case, Step (A) actually reduces to a separation sub-problem on $\mathbf{y}$ only: the goal is to find some extreme ray $\mathbf{u}^e \in \mathcal{P}$ such that $0 < (\mathbf{b} - \mathbf{B}\mathbf{y})^\top \mathbf{u}^e$.

2.2 Benders decomposition with zero flow costs and feasibility cuts

We hereafter focus on the case $\mathbf{c} = \mathbf{0}$ that can arise, for instance, when $\mathbf{c}$ represents zero flow costs in network design or network loading problems (see examples in Section 3.1). When $\mathbf{c} = \mathbf{0}$, all constraints involving $\mathbf{x}$ in the initial program (2.1a)-(2.1d) are only useful to decide on the feasibility of $\mathbf{y}$, because $\mathbf{x}$ has no influence in the objective function (2.1a). As such, the projected cost $\hat{z}$ in the primal-dual LPs (2.2c)-(2.3) can be either 0 when a primal feasible $\mathbf{x}$ exists for the current $\mathbf{y}$ in (2.2c), or ∞ when (2.2c) is infeasible. The goal is thus to find design variables $\mathbf{y}$ that respect the design constraints (2.2b) and that allow $\mathbf{x}$ to receive feasible values in (2.2c). The feasible solutions in (2.2a)-(2.2d) can only have a form $(\mathbf{y}, \hat{z})$ with $\hat{z} = 0$.

Observation 2. *The Benders sub-problem polytope* $\mathcal{P}$ *in* (2.4) *is a pointed polyhedral cone when* $\mathbf{c} = \mathbf{0}$*, as it takes the form* $\mathcal{P} = \left\{\mathbf{u} \geq \mathbf{0} : \mathbf{A}^\top \mathbf{u} \leq \mathbf{0}\right\}$*. This means that* $\mathcal{P}$ *consists only of rays, because* $\mathbf{u} \in \mathcal{P} \implies \alpha\mathbf{u} \in \mathcal{P}, \ \forall \alpha \geq 0$*. The only vertex of* $\mathcal{P}$ *is* $\mathbf{0}$*. This has the following consequences on the Benders reformulation (2.5a)-(2.5d). Since* (2.5b) *is always respected by the only vertex of* $\mathcal{P}$*, (i.e.,* $\mathbf{0}$*), it can be ignored. As such, we solely consider feasibility cuts* (2.5c) *that can be written* $0 \geq (\mathbf{b} - \mathbf{B}\mathbf{y})^\top \mathbf{u}, \forall \mathbf{u} \in \mathcal{P}$.[4] *Such a cut remains unchanged if we replace* $\mathbf{u}$ *by* $\alpha\mathbf{u}$*,* $\forall \alpha \geq 0$*. We can hereafter work with*

$$\mathcal{P}' = \left\{\mathbf{u} \geq \mathbf{0} : \ \mathbf{A}^\top \mathbf{u} \leq \mathbf{0}, \ \mathbf{1}^\top \mathbf{u} = 1\right\}, \tag{2.6}$$

since any ray $\alpha\mathbf{u}$ *of* $\mathcal{P}$ *intersects* $\mathcal{P}'$ *in a unique point* $\mathbf{u}'$ *such that* $0 \geq (\mathbf{b} - \mathbf{B}\mathbf{y})^\top \mathbf{u}' \iff 0 \geq (\mathbf{b} - \mathbf{B}\mathbf{y})^\top \mathbf{u}$.

[4] Using the presentation from [7, § 2.5], this constraint can also be seen as a consequence of the Farkas' lemma: the system of equations $\{\mathbf{A}\mathbf{x} \geq \mathbf{b} - \mathbf{B}\mathbf{y}, \mathbf{x} \geq \mathbf{0}\}$ from (2.2c) has a solution if and only if all $\mathbf{u} \geq 0$ such that $\mathbf{A}^\top \mathbf{u} \leq \mathbf{0}$ satisfy $(\mathbf{b} - \mathbf{B}\mathbf{y})^\top \mathbf{u} \leq 0$ (*i.e.*, if and if all $\mathbf{u} \in \mathcal{P}$ satisfy $0 \geq (\mathbf{b} - \mathbf{B}\mathbf{y})^\top \mathbf{u}$). In other words, there is no hyperplane separating $\mathbf{b} - \mathbf{B}\mathbf{y}$ from the conical hull of the columns of $\mathbf{A}$ and of the negative orthant.

Using the above observation, the Benders reformulation (2.5a)-(2.5d) simplifies to the integer-only master problem below with feasibility cuts only, *i.e.*, the optimality cuts (2.5b) are dropped. Notice that all constraints (2.7c) are associated to vertices of $\mathcal{P}'$, or, equivalently, to rays of $\mathcal{P}$.

$$\min \mathbf{d}^\top \mathbf{y} \tag{2.7a}$$
$$\mathbf{D}\mathbf{y} \geq \mathbf{e} \tag{2.7b}$$
$$0 \geq (\mathbf{b} - \mathbf{B}\mathbf{y})^\top \mathbf{u} \text{ for all } \mathbf{u} \in \mathcal{P}' \tag{2.7c}$$
$$\mathbf{y} \in \mathbb{Z}^n_+ \tag{2.7d}$$

As for the master mixed-integer problem (2.5a)-(2.5d) from the previous Section 2.1, a Benders `Cutting-Planes` algorithm can generate the constraints (2.7c) one by one, using a repeated call to the separation sub-problem $\max\{(\mathbf{b} - \mathbf{B}\mathbf{y})^\top \mathbf{u} : \mathbf{u} \in \mathcal{P}'\}$. It can execute the same steps indicated by Observation 1 (previous page) for a master with both integer and real variables. In fact, all Benders `Cutting-Planes` concepts discussed so far apply perfectly well for this new integer-only master problem (2.7a)-(2.7d). However, a canonical `Cutting-Planes` for (2.7a)-(2.7d) would not allow one to generate feasible solutions $\mathbf{y}$ (upper bounds) along the iterations. We will next present in Section 2.3 an intersection algorithm for this new master problem. This algorithm could actually be extended to the case of non-zero flow costs $\mathbf{c} \neq \mathbf{0}$, with both optimality and feasibility cuts, as described in Appendix B.

2.3 Solving the intersection sub-problem

Definition 1. *(Benders intersection sub-problem) Given ray* $\mathbf{0} \rightarrow \mathbf{r}$*, the intersection sub-problem along* $\mathbf{0} \rightarrow \mathbf{r}$ *asks to find:*

- the minimum $t^* \geq 0$ such that $\mathbf{y} = t^*\mathbf{r}$ is feasible with regards to constraints (2.7c), if such t^* exists;
- an element $\mathbf{u} \in \mathcal{P}'$ for which (2.7c) is satisfied with equality by $t^*\mathbf{r}$, when a t^* defined above exists.

A solution $t^*\mathbf{r}$ calculated by solving the intersection sub-problem is always feasible with regards to constraints (2.7c), but technically not necessarily feasible with regards to the design constraints (2.7b) or to the integrality constraints (2.7d). We can simply overcome this as follows. First, the number of constraints (2.7b) is bounded and it is possible to list them all and pick the minimum value t_D such that $t_D\mathbf{r}$ is feasible with regards to all (2.7b). By executing at the end of the intersection algorithm an assignment $t^* \leftarrow \max(t^*, t_D)$, we simply obtain a solution $t^*\mathbf{r}$ that is feasible with regards to both (2.7b) and (2.7c).

To satisfy the integrality constraint (2.7d), it is possible to round up all components of $t^*\mathbf{r}$, *i.e.*, construct solution $\mathbf{y}^*$ such that $y^*_i = \lceil t^* r_i \rceil \ \forall i \in [1..n]$. In network design, the variables y_i often represent a number of installed transmission facilities and there is no constraint that forbids increasing this number. Under this interpretation, $\mathbf{y}^* \geq t^*\mathbf{r}$ means that $\mathbf{y}^*$ installs more facilities than $t^*\mathbf{r}$; as such, $\mathbf{y}^*$ can accommodate more flow than $t^*\mathbf{r}$, and so, $\mathbf{y}^*$ is also feasible. For an explicit application example, see also Observation 4 at the end of Section 3.3.

Theorem 1. *When the flow costs are* $\mathbf{c} = \mathbf{0}$ *and the demands satisfy* $\mathbf{b} \geq \mathbf{0}$*, the intersection sub-problem is as tractable as the separation sub-problem.*

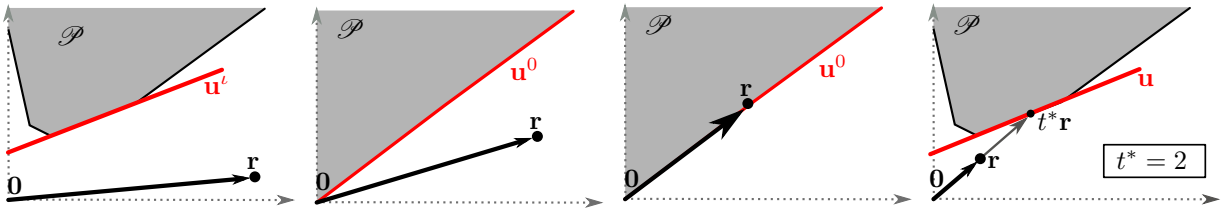
Proof. We will show that the intersection sub-problem requires solving a few linear programs over a polytope of the size of $\mathcal{P}'$, *i.e.*, of the same size as the one solved by the separation sub-problem.

Given $\mathbf{r}$ as input to the intersection sub-problem, we replace $\mathbf{y}$ with $t^*\mathbf{r}$ in (2.7c) and obtain:

$$t^*(\mathbf{B}\mathbf{r})^\top \mathbf{u} \geq \mathbf{b}^\top \mathbf{u}, \ \forall \mathbf{u} \in \mathcal{P}' \tag{2.8}$$

The goal of the intersection sub-problem is to find the minimum $t^* \geq 0$ that satisfies the above family of constraints for the current $\mathbf{r}$. We need to separate two main cases:

(i) a degenerate case in which *no* $t^* \geq 0$ satisfies (2.8) above, *i.e.*, the ray $\mathbf{0} \rightarrow \mathbf{r}$ does not even "touch" the polytope $\mathscr{P}$, see case (i) of Figure 2 next page for an intuitive illustration;



case **(i)**: no intersection, whole ray infeasible

case **(ii.a.1)**: only $t\mathbf{r} = \mathbf{0}$ could be feasible

case **(ii.a.2)**: a zero case, $(\mathbf{Br})^\top \mathbf{u}^0 = \mathbf{b}^\top \mathbf{u}^0 = 0$

case **(ii.b)**: proper intersection, $(\mathbf{Br})^\top \mathbf{u} > 0$

Figure 2: The four cases of ray projection in Theorem 1

(ii) the non-degenerate in which there exists some $t^* \geq 0$ that satisfies (2.8) above.

We first address the degenerate case (i). It arises if there is some $\mathbf{u}^\iota \in \mathcal{P}'$ such that $(\mathbf{Br})^\top \mathbf{u}^\iota \leq 0$ and $\mathbf{b}^\top \mathbf{u}^\iota > 0$. To detect such cases, it is enough to maximize:

$$\max \left\{ \mathbf{b}^\top \mathbf{u} : \ \mathbf{u} \in \mathcal{P}', \ (\mathbf{Br})^\top \mathbf{u} \leq 0 \right\} \tag{2.9}$$

If this LP has an optimal objective value strictly greater than 0 for some $\mathbf{u}^\iota \in \mathcal{P}'$, then there is no $t^* \geq 0$ that respects (2.8) for $\mathbf{u}^\iota$. The intersection sub-problem returns $\mathbf{u}^\iota \in \mathcal{P}'$, but it reports no feasible t^*. A Benders `Cutting-Planes` algorithm can simply deal with this case by adding the valid inequality $(\mathbf{By})^\top \mathbf{u}^\iota \geq \mathbf{b}^\top \mathbf{u}^\iota$ to the current relaxed master ILP associated to (2.7a)-(2.7d), so as to separate the current $\mathbf{r}$ and all its multiples $t\mathbf{r}$, $t \geq 0$.

We now address the non-degenerate case (ii), considering there is *no* $\mathbf{u}^\iota \in \mathcal{P}'$ satisfying the conditions of the above degenerate case (i), *i.e.*, there is *no* $\mathbf{u}^\iota \in \mathcal{P}'$ such that $(\mathbf{Br})^\top \mathbf{u}^\iota \leq 0$ and $\mathbf{b}^\top \mathbf{u}^\iota > 0$. Assuming $\mathcal{P}'$ is not empty (otherwise, (2.7a)-(2.7d) has a limited number of constraints), *all* $\mathbf{u} \in \mathcal{P}'$ respect one of the following, depending on the value of $(\mathbf{Br})^\top \mathbf{u}$:

(ii.a) $(\mathbf{Br})^\top \mathbf{u} \leq 0$ and $\mathbf{b}^\top \mathbf{u} = 0$. Notice that $(\mathbf{Br})^\top \mathbf{u} \leq 0 \implies \mathbf{b}^\top \mathbf{u} = 0$, because both $\mathbf{b}^\top \mathbf{u} > 0$ and $\mathbf{b}^\top \mathbf{u} < 0$ are impossible in this case. Indeed, $\mathbf{b}^\top \mathbf{u} > 0$ can not hold because we ruled out (i) above and $\mathbf{b}^\top \mathbf{u} < 0$ is impossible because all $\mathbf{u} \in \mathcal{P}'$ satisfy $\mathbf{u} \geq 0$ in (2.6) and $\mathbf{b} \geq 0$ by hypothesis.

(ii.b) $(\mathbf{Br})^\top \mathbf{u} > 0$.

The case (ii.a) can be detected by solving (2.9). If the best objective value in (2.9) is exactly 0, then there exists at least a vector $\mathbf{u}^0 \in \mathcal{P}'$ such that $\mathbf{b}^\top \mathbf{u}^0 = 0$ and $(\mathbf{Br})^\top \mathbf{u}^0 \leq 0$. We need to further distinguish between $(\mathbf{Br})^\top \mathbf{u}^0 < 0$ and $(\mathbf{Br})^\top \mathbf{u}^0 = 0$, see also cases (ii.a.1) and (ii.a.2) of Figure 2 for an intuitive illustration. Let us now take $\mathbf{u}^0 = arg \min \left\{ (\mathbf{Br})^\top \mathbf{u} : \ \mathbf{u} \in \mathcal{P}', \ \mathbf{b}^\top \mathbf{u} = 0 \right\}$.

(ii.a.1) If $(\mathbf{Br})^\top \mathbf{u}^0 < 0$, then the constraint (2.7c) or (2.8) defined by $\mathbf{u}^0$ separates $\mathbf{r}$, as well as all $t\mathbf{r}$ with $t > 0$. In this case, the intersection sub-problem returns $\mathbf{u}^0$ to make the `Cutting-Planes` separate $\mathbf{r}$. Since the constraint (2.7c) defined by $\mathbf{u}^0$ separates all $t\mathbf{r}$ with $t > 0$, this is a very strong cut and there is no need to search for other constraints, *i.e.*, the intersection sub-problem algorithm could stop and return $\mathbf{u}^0$, to provide a strong constraint for the next Cutting-Planes iteration.

(ii.a.2) If $(\mathbf{Br})^\top \mathbf{u}^0 = 0$, then all $\mathbf{y} = t\mathbf{r}$ with $t \geq 0$ are feasible with regards to the constraint (2.7c) or (2.8) defined by $\mathbf{u}^0$. The smallest $t^* \geq 0$ value associated to this $\mathbf{u}^0$ is $t^* = 0$. To find vectors $\mathbf{u} \in P'$ that lead to higher t^* values, the intersection algorithm has to continue with the non-degenerate case (ii.b).

We hereafter address the main case (ii.b), *i.e.*, the case of vectors $\mathbf{u} \in \mathcal{P}'$ that can lead to a larger t^* value than the value $t^* = 0$ that might be associated to case (ii.a.2) above. The existence of such $\mathbf{u}$ can be detected by maximizing $(\mathbf{Br})^\top \mathbf{u}$ over all $\mathbf{u} \in \mathcal{P}'$. If the objective value is not strictly positive, case (ii.b) does not exist and the proof is finished in case (ii.a). Otherwise, the set of $\mathbf{u} \in \mathcal{P}'$ satisfying (ii.b) is not empty. In this case, the intersection sub-problem asks to find the minimum t^* that satisfies:

$$t^* \geq \frac{\mathbf{b}^\top \mathbf{u}}{(\mathbf{Br})^\top \mathbf{u}}, \ \forall \mathbf{u} \in P', (\mathbf{Br})^\top \mathbf{u} > 0$$

To determine t^*, one needs to solve:

$$t^* = \max \left\{ \frac{\mathbf{b}^\top \mathbf{u}}{(\mathbf{Br})^\top \mathbf{u}} : \ \mathbf{u} \in \mathcal{P}', \ (\mathbf{Br})^\top \mathbf{u} > 0 \right\}, \tag{2.10}$$

This is a linear-fractional program that can be solved within the same asymptotic running time as a linear program over $\mathcal{P}'$, using the following modelling inspired by the Charnes-Cooper transformation [3]. Writing

$$\overline{\mathbf{u}} = \mathbf{u} \frac{1}{(\mathbf{Br})^\top \mathbf{u}}, \tag{2.11}$$

the linear-fractional program (2.10) above translates into the following pure linear program:

$$t^* = \max \mathbf{b}^\top \overline{\mathbf{u}} \tag{2.12a}$$

$$\mathbf{A}^\top \overline{\mathbf{u}} \leq \mathbf{0} \tag{2.12b}$$

$$(\mathbf{Br})^\top \overline{\mathbf{u}} = 1 \tag{2.12c}$$

$$\overline{\mathbf{u}} \geq \mathbf{0} \tag{2.12d}$$

We now show that any $\mathbf{u} \in P'$ in case (ii.b) is associated to a feasible $\overline{\mathbf{u}} = \frac{\mathbf{u}}{(\mathbf{Br})^\top \mathbf{u}}$ in (2.12a)-(2.12d) such that $\mathbf{u}$ and $\overline{\mathbf{u}}$ have the same objective value, in (2.10) and resp. (2.12a). First, since $(\mathbf{Br})^\top \mathbf{u} > 0$, the value of $\overline{\mathbf{u}}$ is well-defined in (2.11). Furthermore, $\overline{\mathbf{u}}$ satisfies (2.12b) because $\overline{\mathbf{u}}$ is $\mathbf{u}$ divided by a positive value and $\mathbf{A}^\top \mathbf{u} \leq \mathbf{0}$, recall the definition of $\mathcal{P}'$ in (2.6). Finally, the constraint $(\mathbf{Br})^\top \overline{\mathbf{u}} = 1$ follows from the definition of $\overline{\mathbf{u}}$, and so does the last constraint ($\overline{\mathbf{u}}$ is $\mathbf{u} \geq \mathbf{0}$ divided by a positive value). This shows that any $\mathbf{u} \in P'$ leading the *linear-fractional program* (2.10) to $t^* = \frac{\mathbf{b}^\top \mathbf{u}}{(\mathbf{Br})^\top \mathbf{u}}$ can be associated to $\overline{\mathbf{u}} = \frac{\mathbf{u}}{(\mathbf{Br})^\top \mathbf{u}}$ that generates the same t^* value in the above *linear program* (2.12a)-(2.12d).

Conversely, we prove that any feasible solution $\overline{\mathbf{u}}^o$ of above (2.12a)-(2.12d) can be associated to a feasible $\mathbf{u}^o \in \mathcal{P}'$ such that $\mathbf{u}^o$ and $\overline{\mathbf{u}}^o$ have the same objective value in, respectively, (2.10) and (2.12a). This $\mathbf{u}^o$ is given by $\mathbf{u}^o = \frac{\overline{\mathbf{u}}^o}{\alpha}$, choosing a value of $\alpha > 0$ so that $\mathbf{1}^\top \mathbf{u}^o = 1$, to make $\mathbf{u}^o$ satisfy the second (equality) constraint of $\mathcal{P}'$ as defined in (2.6). The first constraint $\mathbf{A}^\top \mathbf{u}^o \leq \mathbf{0}$ of $\mathcal{P}'$ is also satisfied by $\mathbf{u}^o$, because $\mathbf{u}^o$ is $\overline{\mathbf{u}}^o$ divided by some positive value α and $\mathbf{A}^\top \overline{\mathbf{u}}^o \leq \mathbf{0}$ holds in (2.12b). To show that $\mathbf{u}^o \in \mathcal{P}'$ leads (2.10) to the same $t^* = \mathbf{b}^\top \overline{\mathbf{u}}^o$, we write the objective value of $\mathbf{u}^o = \frac{\overline{\mathbf{u}}^o}{\alpha} \in \mathcal{P}'$ in (2.10) as follows:

$$\frac{\mathbf{b}^\top \mathbf{u}^o}{(\mathbf{Br})^\top \mathbf{u}^o} = \frac{\mathbf{b}^\top \frac{\overline{\mathbf{u}}^o}{\alpha}}{(\mathbf{Br})^\top \frac{\overline{\mathbf{u}}^o}{\alpha}} = \frac{\mathbf{b}^\top \overline{\mathbf{u}}^o}{(\mathbf{Br})^\top \overline{\mathbf{u}}^o} = \frac{\mathbf{b}^\top \overline{\mathbf{u}}^o}{1} = t^*, \tag{2.13}$$

where we used $(\mathbf{Br})^\top \overline{\mathbf{u}}^o = 1$, as imposed by (2.12c). This shows that $\mathbf{u}^o = \frac{\overline{\mathbf{u}}^o}{\alpha} \in \mathcal{P}'$ leads (2.10) to the same t^* value as the one achieved by $\overline{\mathbf{u}}^o$ in (2.12a). Also, notice that this $\mathbf{u}^o$ respects condition (ii.b), because $(\mathbf{Br})^\top \mathbf{u}^o = \frac{(\mathbf{Br})^\top \overline{\mathbf{u}}^o}{\alpha} = \frac{1}{\alpha} > 0$.

We still need to address the (degenerate) case of unbounded rays in (2.12a)-(2.12d), *i.e.*, the case in which (2.12a)-(2.12d) contains some ray of the form $\overline{\mathbf{u}} = \overline{\mathbf{u}}' + \beta \mathbf{z}$ (with $\beta \geq 0$) of unbounded objective value. We show below that one can associate such $\mathbf{z}$ to a solution $\mathbf{z}' \in \mathcal{P}'$ that is degenerate as in case (i) above, *i.e.*, a $\mathbf{z}' \in \mathcal{P}'$ such that $\mathbf{b}^\top \mathbf{z}' > 0$ and $(\mathbf{Br})^\top \mathbf{z}' = 0$. This proves that if (2.12a)-(2.12d) contained such unbounded rays, the intersection algorithm would stop in degenerate case (i). As such, if the intersection algorithm arrives at case (ii.b), then such unbounded rays can not exist. The objective value $\mathbf{b}^\top(\overline{\mathbf{u}}' + \beta \mathbf{z})$ can only be unbounded when $\beta \to \infty$ if $\mathbf{b}^\top \mathbf{z} > 0$. Replacing $\overline{\mathbf{u}} = \overline{\mathbf{u}}' + \beta \mathbf{z}$ in (2.12c), we observe $(\mathbf{Br})^\top(\overline{\mathbf{u}}' + \beta \mathbf{z}) = 1 \forall \beta > 0 \implies (\mathbf{Br})^\top \mathbf{z} = 0$. We still need to show that $\mathbf{z} \in \mathcal{P}$: for this, we replace $\overline{\mathbf{u}} = \overline{\mathbf{u}}' + \beta \mathbf{z}$ in (2.12b) and we obtain $\mathbf{A}^\top(\overline{\mathbf{u}}' + \beta \mathbf{z}) \leq \mathbf{0}$. Since this is satisfied by all $\beta > 0$ we need to have $\mathbf{A}^\top \mathbf{z} \leq 0$, *i.e.*, $\mathbf{z}$ has to belong to $\mathcal{P}$ as described in (2.4). As such, $\mathbf{z}$ belongs to $\mathcal{P}$ and satisfies both $\mathbf{b}^\top \mathbf{z} > 0$ and $(\mathbf{Br})^\top \mathbf{z} = 0$, *i.e.*, after an appropriate scaling using a factor $\alpha > 0$ (as in the above paragraph), we obtain $\mathbf{z}' = \frac{\mathbf{z}}{\alpha} \in \mathcal{P}'$ that belongs to the above degenerate case (i). This case would thus be detected by solving the LP (2.9) associated to the case (i). □

2.4 Basic Benders `Cutting-Planes` using the intersection sub-problem

We propose a new Benders `Cutting-Planes` algorithm for (2.7a)-(2.7d), by replacing the separation sub-problem of the standard Benders `Cutting-Planes` (see Observation 1, p. 4) with the intersection sub-problem. At each iteration, the intersection sub-problem leads to one of the cases discussed in Theorem 1:

(i) an $\mathbf{u}^\iota \in \mathcal{P}'$ such that no $\mathbf{y} = t^*\mathbf{r}$ can be feasible with regards to the constraint (2.7c) or (2.8) defined by $\mathbf{u}^\iota$, *i.e.*, the degenerate case in which the ray $\mathbf{0} \rightarrow \mathbf{r}$ does not even "touch" the feasible polytope $\mathscr{P}$. It is thus enough to add a cut (2.7c) defined by $\mathbf{u}^\iota$ to separate all $t\mathbf{r}$ with $t \geq 0$.

(ii.a.1) an $\mathbf{u}^0 \in \mathcal{P}'$ such that $\mathbf{b}^\top\mathbf{u}^0 = 0$ and $(\mathbf{Br})^\top\mathbf{u}^0 < 0$. One has to add to the current relaxed master ILP the constraint (2.7c) associated to $\mathbf{u}^0$, so as to separate all $t\mathbf{r}$ with $t > 0$.

(ii.a.2) an $\mathbf{u}^0 \in \mathcal{P}'$ such that $\mathbf{b}^\top\mathbf{u}^0 = 0$ and $(\mathbf{Br})^\top\mathbf{u}^0 = 0$. This is equivalent to finding $t^* = 0$ (all $t\mathbf{r}$ with $t \geq 0$ are feasible) and it happens only when case (ii.b) does not lead to a higher t^*. The current value of $\mathbf{r}$ can not be separated and the `Cutting-Planes` algorithm finishes by reporting an optimal $\mathbf{r}$.

(ii.b) an $\mathbf{u} \in \mathcal{P}'$ that maximizes the linear-fractional program (2.10) and the associated $t^* = \frac{\mathbf{b}^\top\mathbf{u}}{(\mathbf{Br})^\top\mathbf{u}}$. The resulting $\mathbf{y} = t^*\mathbf{r}$ satisfies all constraints (2.7c) and it can be easily used to determine an upper bound. If $t^* \leq 1$, then $\mathbf{r}$ is feasible, and so, the `Cutting-Planes` algorithm finishes reporting optimal solution $\mathbf{r}$. Otherwise, $\mathbf{r}$ does not satisfy the constraint (2.7c) defined by $\mathbf{u}$. As such, the `Cutting-Planes` process adds to the current relaxed master ILP the cut (2.7c) associated to $\mathbf{u}$, so as to separate $\mathbf{r}$.

After enriching the current (relaxed) master ILP with a new constraint (2.7c) generated as above, the new proposed Benders `Cutting-Planes` algorithm has to (re-)optimize the resulting (relaxed) master ILP, to obtain the next optimal solution (ray $\mathbf{r}$). Optimizing this master *integer* program requires (much) more computing time than the intersection sub-problem algorithm that only solves a few *pure* LPs.

For both the intersection and the separation sub-problems, it is always possible to go beyond the basic Benders `Cutting-Planes` discussed until here. For instance, it is known that the separation sub-problem might often have multiple optimal solutions, associated to different cuts. Finding the strongest cuts in this case might have an important effect upon the efficiency. We refer the reader to [9, 5, 4, 6, 7] for interesting strategies of generating effective Benders cuts (referred to as strengthening methods, accelerating schemes, or separation techniques to generate non-dominated constraints, "simultaneous" Benders cuts, disjoint Benders cuts, etc.). For instance, one of the earliest approaches dating from the 1980s consists of generating constraints that are Pareto-optimal, *i.e.*, not dominated by other Benders cuts [9]. Such strategies have been designed for the separation sub-problem, but they can also be used to accelerate `Cutting-Planes` algorithms based on the intersection sub-problem. However, experiments suggest that the intersection sub-problem has relatively few optimal solutions, less than the separation sub-problem. We discuss further related ideas in Section 2.5 below, along with other enhancement techniques for both Benders `Cutting-Planes`.

2.5 Advanced `Cutting-Planes` using separations and intersections

2.5.1 The intersection sub-problem can find stronger normalized cuts by construction

Let us first introduce the central idea of this section in the context of a general LP (1.1), recalled below for the reader's convenience.

$$\min\left\{\mathbf{d}^\top\mathbf{y} : \mathbf{u}^\top\mathbf{y} \geq b,\ \forall(\mathbf{u}, b) \in \mathcal{C},\ \mathbf{y} \in \mathbb{Z}^n_+\right\} = \min\left\{\mathbf{d}^\top\mathbf{y} : \mathbf{y} \in \mathscr{P},\ \mathbf{y} \in \mathbb{Z}^n_+\right\}$$

We start with a simple example (with $n = 3$ variables) that nevertheless captures the essence of this section. Consider, for instance, that $\mathbf{r} = [1\ 1\ 1]^\top$ is the current optimal solution at a given `Cutting-Planes` iteration and that one has to choose between the following (Benders) cuts:

(1) $4y_1 + 6y_2 + 8y_3 >= 20$, *i.e.*, $\mathbf{u}^1 = [4\ 6\ 8]^\top$ and $b^1 = 20$;

(2) $y_1 + 2y_2 + y_3 >= 5$, *i.e.*, $\mathbf{u}^2 = [1\ 2\ 1]^\top$ and $b^2 = 5$.

The standard separation problem would choose constraint (1) by solving $\max_{(\mathbf{u},b)\in\mathcal{C}} b - \mathbf{u}^\top\mathbf{r}$ as in (1.2), notice that $20-4-6-8 = 2 > 1 = 5-1-2-1$. The intersection sub-problem would choose (2) by solving $\max_{(\mathbf{u},b)\in\mathcal{C}} \frac{b}{\mathbf{u}^\top\mathbf{r}}$ as in (1.3), notice that $\frac{20}{4+6+8} = 1+\frac{1}{9} < 1+\frac{1}{4} = \frac{5}{1+2+1}$. We can easily argue that constraint (2) is indeed stronger than (1), more by violated by $\mathbf{r} = [1\ 1\ 1]$. For this, it is enough to normalize constraints (1) and (2), to make them both have the same right-hand side value, allowing us to make an unbiased comparison. We obtain the following normalized cuts, completely equivalent to above (1) and resp. (2).

(1') $0.2y_1 + 0.3y_2 + 0.4y_3 >= 1$, *i.e.*, $\mathbf{u}^1 = [0.2\ \ 0.3\ \ 0.4]^\top$ and $b = 1$;

(2') $0.2y_1 + 0.4y_2 + 0.2y_3 >= 1$, *i.e.*, $\mathbf{u}^2 = [0.2\ \ 0.4\ \ 0.2]^\top$ and $b = 1$.

When comparing these normalized constraints, we can say that $(2')$ dominates $(1')$ because $0.2+0.3+0.4 > 0.2+0.4+0.2$, and so, $(2')$ is more violated than $(1')$ by the current optimal solution $\mathbf{r} = [1\ 1\ 1]^\top$. Both the separation and the intersection sub-problem would choose $(2')$ in this case.

More generally, if all cuts were normalized, the separation and the intersection sub-problem would both return the same cut – assuming $\mathbf{u}^\top\mathbf{r} > 0\ \forall(\mathbf{u},b)\in\mathcal{C}$. Indeed, if the right-hand side is always fixed to $b = 1$, both the separation and the intersection sub-problem reduce to finding $\min_{(\mathbf{u},1)\in\mathcal{C}} \mathbf{u}^\top\mathbf{r}$.

Let us focus on the more general case in which the cuts are not all normalized. In this case, we can say that the intersection sub-problem first normalizes all cuts and then returns the most violated one. Indeed, one could determine $\max_{(\mathbf{u},b)\in\mathcal{C}} \frac{b}{\mathbf{u}^\top\mathbf{r}}$ by normalizing all constraints followed by minimizing $\min_{(\mathbf{u},1)\in\overline{\mathcal{C}}} \mathbf{u}^\top\mathbf{r}$, where $\overline{\mathcal{C}}$ is the set of normalized constraints. Solving the intersection sub-problem is thus equivalent to normalizing all constraints and separating (by minimizing $\min_{(\mathbf{u},1)\in\overline{\mathcal{C}}} \mathbf{u}^\top\mathbf{r}$). We can easily argue that it makes most sense to compare two cuts only when they are normalized. This idea has been implicitly used since 1981, when [9, § 2] defined the domination relation by comparing cuts with the same variable-free term (right-hand side value b in our terminology). One can say that the intersection sub-problem requires finding *the most violated normalized cut*; this is actually the strongest constraint in the sense that it is more violated than other constraints *with the same right-hand value*.

On the other hand, experiments suggest that the weak point of the new Benders `Cutting-Planes` is the degenerate case (i) from Theorem 1, *i.e.*, the ray $\mathbf{r}$ does not even "touch" the feasible polytope $\mathscr{P}$. This can only happen when there is a constraint $\mathbf{u}_\iota^\top\mathbf{y} \geq b_\iota$ such that $\mathbf{u}_\iota^\top\mathbf{r} \leq 0$ and $b_\iota > 0$. A very unfortunate case example is illustrated in Figure 6. It can arise, for instance, when one has to choose between (a) $3y_1 + 2y_2 + 2y_3 \geq 10$ and (b) $50y_1 \geq 1$ for $\mathbf{r} = [0\ 1\ 1]^\top$. The intersection sub-problem has to return constraint (b) as a degenerate case, *i.e.*, the constraint (b) makes the whole ray infeasible, because all points $t\mathbf{r} = [0\ t\ t]^\top$ with $t \geq 0$ violate (b). The separation sub-problem returns the cut (a) which is stronger than (b) in the sense that one has to increase r_1 to $r_1 = 2$ to satisfy (a), why a small value of $r_1 = 0.02$ is enough to satisfy (b). We can no longer take profit from the above ideas on normalized cuts, because the intersection algorithm does no longer minimize a ratio, but it has to maximize an LP of the form (2.9) to find constraints like $\mathbf{u}_\iota^\top\mathbf{y} \geq b_\iota$ above. We present in the next subsection a smoothing strategy that can reduce such limitations and also overcome other drawbacks.

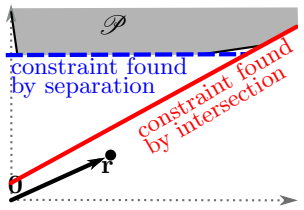


Figure 3: An unfortunate case for the intersection sub-problem (degenerate case).

2.5.2 Accelerating the convergence: smoothed solutions for separation and intersection

A frequent drawback of `Cutting-Planes` algorithms is that the convergent progress can be characterized by strong oscillations of the current optimal solution along the iterations, which can seriously slow down the convergence. If the current optimal solution is taken as query point (*i.e.*, by calling the separation sub-problem on it), the `Cutting-Planes` algorithm is also referred to as the Kelley's method. The query points generated along the iterations can be far from each other, and also quite far from the feasible area.

As [10] put it, this algorithm has a lack of stability that had been noted for a long time, with intermediate "solutions possibly moving dramatically after adding cutting planes". Experiments suggest that our Benders `Cutting-Planes` algorithms do suffer from such issues, *i.e.*, the optimal solution at a given iteration might share very few features (selected edges) with the optimal solution at the previous iteration.

An approach that can (partially) overcome this drawback consists of "solution smoothing", *i.e.*, instead of applying the separation or the intersection sub-problem on the current optimal solution at each iteration, we apply it on a smoothed solution. The query point is a thus smoothed solution that can be obtained, for instance, by taking the midpoint between the current optimal solution and the previous optimal solution. This solution smoothing approach is also very popular as a stabilization method in `Column Generation`, where the dual optimal solutions exhibit strong oscillations along the iterations. At least in this context, the solution smoothing approach can address at the same time "oscillations, tailing-off and degeneracy drawbacks" – see [11] for arguments on this and further details on (dual) solution smoothing.

We propose to use this smoothing technique *both* for the standard and the new Benders `Cutting-Planes` algorithms. For the standard algorithm, we first apply the separation sub-problem on a smoothed solution (the midpoint between the current and the previous optimal solution). If the resulting cut separates the current optimal solution, we say the smoothed cut is successful (a hit). Otherwise, the smoothed cut is unsuccessful and we need to call a second separation on the current optimal solution.

Regarding the new `Cutting-Planes` algorithm, we consider that the search progress consists of two phases: a degenerate phase in which most rays (current optimal solutions) do not even "touch" the feasible polytope $\mathscr{P}$ (*i.e.*, they are in the degenerate case (i) of Theorem 1) and a normal phase in which the rays $\mathbf{r}$ usually intersect the feasible polytope $\mathscr{P}$ in points of the form $t^*\mathbf{r}$. We consider that the search is in the degenerate phase as long as the best upper bound is more than twice the current lower bound.

At each iteration of the degenerate phase, we determine a smoothed solution as for the standard Benders `Cutting-Planes` case, *i.e.*, take the midpoint between the current optimal solution $\mathbf{r}$ and the optimal solution at the previous iteration. As argued in Section 2.5.1 above, this degenerate phase is the weak point of the new Benders `Cutting-Planes`, because the cuts generated by intersection are not particularly strong when the rays do not "touch" the feasible polytope $\mathscr{P}$. To overcome this, the advanced version of the new `Cutting-Planes` algorithm first applies the standard separation sub-problem on the smoothed solution. If the resulting cut shows that the ray $\mathbf{0} \rightarrow \mathbf{r}$ does not "touch" $\mathscr{P}$, we then consider that the smoothed cut is successful (a hit). Otherwise, the smoothed cut is unsuccessful and we need to call the intersection sub-problem on the current ray $\mathbf{r}$. By using a separation sub-problem during this degenerate phase, one can say that the advanced version of the new Benders `Cutting-Planes` algorithm actually combines the intersection and the separation sub-problems.

During the normal (non-degenerate) phase, the smoothing technique can use the feasible primal solutions generated by solving intersection sub-problems at previous iterations. As such, the query point is defined as the the midpoint $\mathbf{r}_m$ between the current optimal solution $\mathbf{r}$ and the best (non-rounded) feasible solution discovered so far. At each iteration, we fist solve the intersection sub-problem on this query point $\mathbf{r}_m$. If the returned cut separates the current optimal solution $\mathbf{r}$, we consider the smoothed intersection is successful (a hit) and we no longer apply the intersection sub-problem on $\mathbf{r}$. Otherwise, the smoothed cut is unsuccessful and we need to call a second intersection sub-problem on $\mathbf{r}$. The use of the best feasible solution to define the query point is reminiscent of centralization methods (or centering schemes), in which one uses more interior solutions as query points – *e.g.*, see references on the analytic-center `Cutting-Planes` method in [10, 11].

2.5.3 Practical acceleration of both methods by avoiding the hardest master ILPs

The most critical computational step of both Benders `Cutting-Planes` is the *iterative* resolution of the (relaxed) master *integer* LP associated to (2.7a)-(2.7d). While the computational effort of solving this ILP does depend on the number of generated constraints, experiments suggest that certain relatively small master ILPs can still require a prohibitively-long computing time. This can occasionally happen in both Benders `Cutting-Planes` algorithms, whenever they produce a particularly difficult combination of generated constraints in the master ILP, so that the `Cplex` ILP solver can remain blocked virtually indefinitely. To

avoid this, we propose to stop the ILP solver if it exceeds a certain CPU time threshold,[5] and let the `Cutting-Planes` continue with the best integer solution $\mathbf{y}_{nopt}$ found by the ILP solver so far, *i.e.*, $\mathbf{y}_{nopt}$ is a sub-optimal solution of the current relaxed master. However, if the (separation or intersection) sub-problem separates $\mathbf{y}_{nopt}$, the `Cutting-Plane` algorithm can add a new constraint, (re-)optimize the resulting master ILP and continue as usually. The fact that $\mathbf{y}_{nopt}$ was sub-optimal does not influence the correctness of (the continuation of) the `Cutting-Planes`. By adding a new constraint to separate $\mathbf{y}_{nopt}$, the resulting master ILP can become reasonably-difficult again, "unblocking" the `Cutting-Plane` process.

If the above $\mathbf{y}_{nopt}$ can not be separated, the Benders `Cutting-Planes` can *not* stop and report $\mathbf{y}_{nopt}$ as an optimal solution, simply because $\mathbf{y}_{nopt}$ is by construction sub-optimal for the current relaxed master. For this case, we propose to multiply the above CPU time threshold by 100 and try again to solve the current (prohibitively-hard) master ILP. If this new larger time limit is enough to solve the master ILP, the Benders `Cutting-Planes` algorithm can continue as usually. Otherwise, the `Cutting-Planes` algorithm stops and we consider that it can not solve the instance at issue. Generally speaking, experiments suggest that even if one multiplied the threshold by 1000, the ILP solver could still fail.

3 An application example

3.1 The primal integer linear model

Suppose one needs to install multiple transmission facilities – such as cables or other telecommunication links – over the edges of a graph (telecommunication network) $G = (V, E)$. We consider a source (origin) O and a set of destination terminals T with $O \notin T$: the goal is to construct a least-cost set of links that allow a multicast flow $\mathbf{x}$ to pass from O to T. In other words, we ask to minimize the total cost of the mounted links $\mathbf{y}$ needed to accommodate a required *one-to-many* flow.

As argued in [14, 8], the flow cost can be ignored in many computer networks: this is realistic when there is no volume-based cost for using installed telecommunication links (*e.g.*, TCP/IP Ethernet cables). However, there is a fixed charge d_{ij} incurred for leasing a communication link from a telecommunication carrier or for installing a private link from i to j. The network design problems with zero flows cost have also been referred to as network *loading* problems [8, 7], in the sense that one has to load transmission facilities (*e.g.*, cables) that can carry flow at no cost along the edges, see also the beginning of [8, §1]. Zero flow costs can also arise in other applications besides computer networks, for instance, in networks of electric lines or in networks of watter supply pipes. It is indeed reasonable to assume that the volume-based cost for using an electric line or a water pipe is insignificant compared to the installation or construction cost.

Our model uses decision variables y_{ij} to indicate the number of installed links between i and j, each of bandwidth (capacity) b_{wd}; variables x_{ij} represent the flow from i to j. Notice that d_{ij} and y_{ij} are undirected variables associated to undirected edges $\{i, j\} \in E$, *i.e.*, we can use the convention $d_{ij} = d_{ji}$ and $y_{ij} = y_{ji}$. In contrast, the flow variables x_{ij} are directed. The following model is an adaptation of (21)-(25) from [4], of (10)-(14) from [5], of (1)-(4) from [7], of (1)-(6) from [6], or of (1)-(3) from [8].

$$\min \sum_{\{i,j\}\in E} d_{ij} y_{ij} \tag{3.1a}$$

$$\sum_{\{i,j\}\in E} x_{ji} - \sum_{\{i,j\}\in E} x_{ij} \geq 0, \ \forall i \notin T \cup \{O\} \tag{3.1b}$$

$$\sum_{\{i,j\}\in E} x_{ji} - \sum_{\{i,j\}\in E} x_{ij} \geq b_i, \ \forall i \in T \tag{3.1c}$$

$$b_{\text{wd}} y_{ij} - x_{ij} - x_{ji} \geq 0, \ \forall \{i, j\} \in E, \ i < j \tag{3.1d}$$

$$y_{ij} \in \mathbb{Z}_+, \ x_{ij}, x_{ji} \geq 0, \ \forall \{i, j\} \in E, \ i < j \tag{3.1e}$$

[5] We use $\left\lfloor \frac{|E|}{120} \right\rfloor + 1$ seconds for the ILP solver of `Cplex 12.6`, where $|E|$ is the number of edges of the underlying graph.

The above model replaced the classical flow conservation equality constraints (see, *e.g.*, (22) of [4] or (1) of [7]) with *inequalities* (3.1b)-(3.1c). For instance, (3.1b) states that the flow entering i has to be *greater than or equal to* the flow exiting i. This is weaker than an equality constraint, but a feasible solution $(\mathbf{y}, \mathbf{x})$ that satisfies a constraint (3.1b) or (3.1c) without equality can be transformed into a feasible solution that satisfies it with equality. For this, it is enough to decrease the flow $\sum_{\{i,j\}\in E} x_{ji}$ entering in i, by decreasing any x_{ji} terms. We prefer to use these inequality flow constraints, because the goal is to make the above LP better fit the model (2.1a)-(2.1d) and Theorem 1. We do not add any constraint (3.1b) or (3.1c) for $i = O$, because there is no flow conservation at the source. The flow values x_{Oj} with $\{O, j\} \in E$ can become as large as necessary, and there is no flow x_{jO} entering O because $O \notin T$.

Finally, constraints (3.1d) ensure that the total traffic on each link is bounded by the total installed bandwidth, *i.e.*, the number of installed links multiplied by the bandwidth b_{wd} of an individual link. This type of inequality arises, for instance, in (3) of [14] or in (2) of [8].

3.2 Constructing $\mathcal{P}'$ and the Benders reformulation model

The variables of the Benders sub-problem polytope $\mathcal{P}$ The inequalities (3.1b)-(3.1d) are instantiations of the general constraints (2.1c). Using the Benders decomposition approach from Section 2.1, these constraints can be used inside an inner LP of the form (2.2c). By dualizing this inner LP as when we constructed (2.3)-(2.4), we first obtain the variables of the Benders sub-problem polytope $\mathcal{P}$ from (2.4):

(a) $|V| - 1$ dual variables $u_i \geq 0$ associated to (3.1b)-(3.1c). There are $|V| - 1$ such dual variables (primal constraints), because there is no constraint (3.1b) or (3.1c) for the source O.

(b) $|E|$ dual variables $u_{ij} \geq 0$ associated to (3.1d). For each edge $\{i, j\}$ with $i < j$, we define a unique dual variable u_{ij};

Summing up above (a) and (b), we obtain that the vectors $\mathbf{u}$ of polytope $\mathcal{P}$ from (2.4) have size $|E| + |V| - 1$.

The constraints of the Benders sub-problem polytope $\mathcal{P}$ There are $2|E|$ constraints in $\mathcal{P}$, because each edge $\{i, j\} \in E$ is associated to two primal variables x_{ij} and x_{ji}. The dual constraints associated to the columns of x_{ij} and resp. x_{ji} (with $i < j$) are (3.2a) and resp. (3.2b) below, constructing the polytope $\mathcal{P}$:

$$\mathcal{P} \begin{cases} -u_{ij} - u_i + u_j \leq 0 \quad \forall \{i, j\} \in E,\ i < j & \text{(3.2a)} \\ -u_{ij} - u_j + u_i \leq 0 \quad \forall \{i, j\} \in E,\ i < j & \text{(3.2b)} \\ \mathbf{u} \geq \mathbf{0}, & \text{(3.2c)} \end{cases}$$

where we use the convention that if i (resp. j) equals O then the term u_i (resp. u_j) vanishes in (3.2a)-(3.2b). The first (resp. second) constraint corresponds to column x_{ij} (resp. x_{ji}). The argument justifying the first constraint is the following: (i) $-u_{ij}$ comes from the $-x_{ij}$ term of (3.1d), (ii) $-u_i$ comes from the $-x_{ij}$ term (flow exiting i) of either (3.1b) or (3.1c) defined by i, (iii) $+u_j$ comes from the x_{ij} term (flow *entering* j) of (3.1b) or (3.1c) defined by j. The second constraint follows from an analogous argument.

Notice that the resulting polytope $\mathcal{P}$ defined by (3.2a)-(3.2c) above has the particular structure described in Observation 2 (p. 4), *i.e.*, it is a pointed polyhedral cone consisting only of rays (except vertex $\mathbf{0}$). Then, the definition of $\mathcal{P}'$ follows immediately by imposing $\mathbf{1}^\top \mathbf{u} = 1$ as stated by (2.6).

The Benders reformulation model We now reformulate (3.1.a)-(3.1e) to obtain a *Benders reformulation* of the general form (2.7a)-(2.7d). In fact, we will instantiate (2.7a)-(2.7d) to our application, using a similar approach as in Section 2.2. Recall that (2.7c) actually states that the dual objective function $(\mathbf{b} - \mathbf{B}\mathbf{y})^\top \mathbf{u}$ over all $\mathbf{u} \in \mathcal{P}'$ (or equivalently $\mathbf{u} \in \mathcal{P}$) has to be at maximum 0. We can generate a similar constraint for our application. For this, first notice that the variables u_i with $i \in T$ have dual objective function coefficient b_i, because constraints (3.1c) have a right-hand side value of b_i; variables u_i with $i \notin T \cup \{O\}$ have dual objective function coefficient 0 because of the right-hand 0 in (3.1b). Similarly, variables u_{ij} have dual objective function coefficient $-b_{\text{wd}} y_{ij}$, by moving $b_{\text{wd}} y_{ij}$ in the right-hand side of (3.1d). Since $\mathcal{P}$ contains

only rays (except $\mathbf{0}$), the following inequality needs to hold to ensure that the dual objective function over $\mathbf{u} \in \mathcal{P}$ in (2.3) is at maximum 0 as in (2.7c).

$$0 \geq -\sum_{\{i,j\}\in E} b_{\text{wd}} y_{ij} u_{ij} + \sum_{i\in T} b_i u_i$$

A similar constraint was derived in [5, (5)] for the (generalized) case of multi-commodity flows. Also, the constraints (3.2a)-(3.2c) defining our polytope $\mathcal{P}$ are very similar to the constraints (4) of [5].

The main Benders reformulation model (2.7a)-(2.7d) becomes:[6]

$$\min \mathbf{d}^\top \mathbf{y} \tag{3.3a}$$

$$0 \geq -\sum_{\{i,j\}\in E} b_{\text{wd}} y_{ij} u_{ij} + \sum_{i\in T} b_i u_i, \forall \mathbf{u} \in \mathcal{P}' \tag{3.3b}$$

$$\mathbf{y} \in \mathbb{Z}^n_+ \tag{3.3c}$$

After determining the optimal solution $\mathbf{y}$ at the current iteration, a `Cutting-Planes` algorithm searches for some $\mathbf{u}' \in \mathcal{P}'$ (*i.e.*, an $\mathbf{u}'$ satisfying (3.2a)-(3.2c) and $\mathbf{1}^\top \mathbf{u}' = 1$) that maximizes $-\sum_{\{i,j\}\in E} b_{\text{wd}} y_{ij} u'_{ij} + \sum_{i\in T} b_i u'_i$, *i.e.*, it tries to separate the current $\mathbf{y}$, so as to add a new constraint to the current relaxed master. The main general `Cutting-Planes` steps are the same as those described in Observation 1 (p. 4).

3.3 Solving the intersection sub-problem

To solve the intersection sub-problem along some $\mathbf{0} \to \mathbf{r}$ in the above Benders reformulation (3.3a)-(3.3c), one has to replace $\mathbf{y} = t^*\mathbf{r}$ in (3.3b) and then find the minimum $t^* \geq 0$ that makes the following (particularization of (2.8)) hold:

$$t^* \cdot \sum_{\{i,j\}\in E} b_{\text{wd}} r_{ij} u_{ij} \geq \sum_{i\in T} b_i u_i \ \forall \mathbf{u} \in \mathcal{P}'. \tag{3.4}$$

Observation 3. *(degenerate case) Following Theorem 1, we first separate the degenerate case (i) in which there is no $t^* \geq 0$ for which the above (3.4) holds. This can only happen if there is some $\mathbf{u}^\iota \in \mathcal{P}'$ such that $\sum_{\{i,j\}\in E} b_{wd} r_{ij} u^\iota_{ij} = 0$ and $\sum_{i\in T} b_i u^\iota_i > 0$. Such a case can be detected by solving the following LP, an instantiation of (2.9) from Theorem 1:*

$$\max \left\{ \textstyle\sum_{i\in T} b_i u_i : \ \mathbf{u} \in \mathcal{P}', \ \sum_{\{i,j\}\in E} b_{wd} r_{ij} u_{ij} = 0 \right\}. \tag{3.5}$$

If the best objective value of this LP is strictly greater than 0 for some $\mathbf{u}^\iota \in \mathcal{P}'$, the proposed `Cutting-Planes` algorithm adds the constraint $\sum_{\{i,j\}\in E} b_{wd} y_{ij} u^\iota_{ij} \geq \sum_{i\in T} b_i u^\iota_i$, so as to separate all $t\mathbf{r}$ with $t \geq 0$ from the feasible area of (3.3a)-(3.3c). Also notice that $\sum_{\{i,j\}\in E} b_{wd} r_{ij} u_{ij} \geq 0$ is always satisfied during the Benders `Cutting-Planes`, because $\mathbf{u} \geq 0$ holds in (3.2c), $\mathbf{r} \in \mathbb{Z}^n_+$ is a non-negative optimal solution of a relaxed master associated to (3.3a)-(3.3c), and b_{wd} is a non-negative bandwidth.

We now address the particular case (ii.a) of Theorem 1, that can be detected if the maximum of the above LP (3.5) is exactly 0. First, the (sub-)case (ii.a.1) can not arise, because $\sum_{\{i,j\}\in E} b_{\text{wd}} r_{ij} u_{ij} \geq 0 \ \forall \mathbf{u} \in \mathcal{P}'$ as explained above. We are left with the particular case (ii.a.2) in which there is some $\mathbf{u}^0 \in \mathcal{P}'$ such that $\sum_{\{i,j\}\in E} b_{\text{wd}} r_{ij} u^0_{ij} = 0$ and $\sum_{i\in T} b_i u^0_i = 0$. This $\mathbf{u}^0$ allows any $t^* \geq 0$ to be feasible in (3.4), yielding $t^* = 0$. As such, the constraint (3.3b) associated to this $\mathbf{u}^0$ does not generally lead to the strongest cut in (3.3a)-(3.3c). If it does, the `Cutting-Planes` algorithm can stop and report that $\mathbf{r}$ is an optimal solution.

[6] All next sums of y or u terms over edges $\{i,j\} \in E$ use the convention $i < j$. Whenever we refer to variables y_{ij}, u_{ij} or r_{ij}, we assume that $i < j$ holds, see also the arguments from the 6$^{\text{th}}$ paragraph of [4, §3].

As in Theorem 1, after separating the above particular cases, we can focus on elements $\mathbf{u} \in \mathcal{P}'$ that respect $\sum_{\{i,j\}\in E} b_{\text{wd}} r_{ij} u_{ij} > 0$, *i.e.*, the main case (ii.b). To find t^* in this case, one needs to solve the following linear-fractional program, a particularization of (2.10):

$$t^* = \max \left\{ \frac{\sum_{i\in T} b_i u_i}{\sum_{\{i,j\}\in E} b_{\text{wd}} r_{ij} u_{ij}} : \ \mathbf{u} \in \mathcal{P}', \ \sum_{\{i,j\}\in E} b_{\text{wd}} r_{ij} u_{ij} > 0 \right\}, \tag{3.6}$$

We can apply the following Charnes-Cooper transformation as in (2.11)

$$\overline{\mathbf{u}} = \mathbf{u} \frac{1}{\sum_{\{i,j\}\in E} b_{\text{wd}} r_{ij} u_{ij}}, \tag{3.7}$$

so as to translate (3.6) into:

$$t^* = \max \mathbf{b}^\top \overline{\mathbf{u}} = \textstyle\sum_{i\in T} b_i \overline{u}_i \tag{3.8a}$$

$$- \overline{u}_{ij} - \overline{u}_i + \overline{u}_j \leq 0 \ \forall \{i,j\} \in E, \ i < j \ ^7 \tag{3.8b}$$

$$- \overline{u}_{ij} - \overline{u}_j + \overline{u}_i \leq 0 \ \forall \{i,j\} \in E, \ i < j \ ^7 \tag{3.8c}$$

$$\sum_{\{i,j\}\in E} b_{\text{wd}} r_{ij} \overline{u}_{ij} = 1 \tag{3.8d}$$

$$\overline{\mathbf{u}} \geq \mathbf{0} \tag{3.8e}$$

As in Theorem 1, any feasible $\mathbf{u} \in \mathcal{P}'$ from (3.6) is transformed by (3.7) into a feasible $\overline{\mathbf{u}}$ in (3.8a)-(3.8e) such that $\mathbf{u}$ and $\overline{\mathbf{u}}$ have the same objective value in (3.6) and resp. (3.8a). To show $\overline{\mathbf{u}}$ is feasible, we first notice that since $\mathbf{u} \in \mathcal{P}'$ satisfies (3.2a)-(3.2b), then $\overline{\mathbf{u}} = \mathbf{u}/\beta$ (with $\beta = \sum_{\{i,j\}\in E} b_{\text{wd}} r_{ij} u_{ij} > 0$) satisfies (3.8b)-(3.8c). Secondly, $\mathbf{u} \geq \mathbf{0}$ from (3.2c) leads to $\overline{\mathbf{u}} = \mathbf{u}/\beta \geq 0$, *i.e.*, constraint (3.8e). Finally, constraint (3.8d) follows from the definition of $\overline{\mathbf{u}}$ in (3.7). It is not hard to check now that the objective value of $\mathbf{u}$ in (3.6) is the same as that of $\overline{\mathbf{u}}$ in (3.8a).

Conversely, any optimal solution $\overline{\mathbf{u}}^o$ of (3.8a)-(3.8e) above[8] can be associated to a feasible $\mathbf{u}^o \in P'$ by applying a scaling of the form $\mathbf{u}^o = \frac{\overline{\mathbf{u}}^o}{\alpha}$, choosing an $\alpha > 0$ such that $\mathbf{1}^\top \mathbf{u}^o = 1$, so as to make $\mathbf{u}^o$ belong to $\mathcal{P}'$. This $\overline{\mathbf{u}}^o$ leads (3.8a) to the same t^* value as the one associated to $\mathbf{u}^o$ in (3.6). This comes from the fact that $\sum_{\{i,j\}\in E} b_{\text{wd}} r_{ij} \overline{u}^o_{ij} = 1$ as stated in (3.8d), so that $\sum_{i\in T} b_i \overline{u}^o_i = \frac{\sum_{i\in T} b_i \overline{u}^o_i}{\sum_{\{i,j\}\in E} b_{\text{wd}} r_{ij} \overline{u}^o_{ij}} = \frac{\sum_{i\in T} b_i u^o_i}{\sum_{\{i,j\}\in E} b_{\text{wd}} r_{ij} u^o_{ij}}$. Finally, $\mathbf{u}^o$ satisfies the last constraint of (3.6), because $\sum_{\{i,j\}\in E} b_{\text{wd}} r_{ij} \overline{u}^o_{ij} = 1 > 0$ and α is a strictly positive value.

After solving the LP (3.8a)-(3.8e), the intersection algorithm returns the hit point $\mathbf{y} = t^*\mathbf{r}$ that satisfies with equality the constraint (3.3b) or (3.4) associated to $\mathbf{u}^o$. If $t^* > 1$, one can separate $\mathbf{r}$ by adding the constraint (3.3b) defined by $\mathbf{u}^o$ to the relaxed master associated to (3.3a)-(3.3c). After adding such a constraint, the `Cutting-Planes` algorithm re-optimizes the resulting master enriched with one constraint more, so as to find a new current optimal solution $\mathbf{r}$. The `Cutting-Planes` based on the intersection sub-problem was discussed in greater detail in Section 2.4 (basic version) and in Section 2.5 (advanced version). All results from Section 2.5 on the advanced `Cutting-Planes` algorithm apply perfectly well to the models from this section, and so, we will use this advanced algorithm for most numerical tests in the next section.

Observation 4. *The solution $\mathbf{y} = t^*\mathbf{r} \geq \mathbf{0}$ returned by the intersection algorithm satisfies all constraints* (3.3b)*, but not necessarily the integrality of $\mathbf{y}$ from* (3.3c)*. To obtain a solution $\mathbf{y}'$ that does respect all constraints* (3.3b)-(3.3c)*, it is enough to set $y'_{ij} = \lceil y_{ij} \rceil$, $\forall \{i,j\} \in E, \ i < j$. This rounded $\mathbf{y}'$ satisfies the constraint* (3.3b) *defined by any $\mathbf{u} \in \mathcal{P}'$, because the following is true by virtue of $\mathbf{u} \geq \mathbf{0}$ and $b_{wd} > 0$:*

$$0 \geq - \sum_{\{i,j\}\in E} b_{wd} y_{ij} u_{ij} + \sum_{i\in T} b_i u_i \geq - \sum_{\{i,j\}\in E} b_{wd} y'_{ij} u_{ij} + \sum_{i\in T} b_i u_i.$$

[7] If i or j is O, the associated term $\overline{u}_O$ vanishes, recall we used the same convention when we defined $\mathcal{P}$ in (3.2a)-(3.2c). For instance, if $j = O$, constraints (3.8b)-(3.8c) become $-\overline{u}_{ij} - \overline{u}_i, -\overline{u}_{ij} + \overline{u}_i \leq 0$.

[8] If (3.8a) is unbounded, $t^*\mathbf{r}$ is infeasible for any $t^* \geq 0$. This case would be detected as the degenerate case (i) described by Observation 3 (p. 13), see the last paragraph of Theorem 1 for a proof.

One can obtain the same result using the primal constraints (3.1b)-(3.1e). *The solution* $\mathbf{y}'$ *allows all required flow to pass in* (3.1b)-(3.1e), *because* $\mathbf{y}'$ *imposes even less constraints than* $\mathbf{y}$ *on the variables* $\mathbf{x}$ *in* (3.1d).

4 Numerical Results: Basic and Advanced Benders `Cutting-Planes`

We start out with a brief Section 4.1 that evaluates the intersection sub-problem on the basic Benders `Cutting-Planes` from Section 2.4, using a few small instances that can generally be solved in less than one minute. We will then continue in Section 4.2 with the advanced accelerated `Cutting-Planes` from Section 2.5, providing statistical results over larger, harder and more varied instances. We will finish in Section 4.3 with an evaluation of the potential integration of the proposed new upper bounds in a `Branch-and-Bound` algorithm based on a linear relaxation.

4.1 Basic `Cutting-Planes` based on the intersection sub-problem

We here evaluate a basic `Cutting-Planes` in which we only replace the separation sub-problem with the intersection sub-problem. We generated 20 instances of 4 different sizes, with $|E|$ ranging from 150 to 220 and $|V|$ ranging from 25 to 50. For each size, we consider 5 random graphs, identified by an *id* from 1 to 5. The demand b_i for each vertex (sink) $i \in V \setminus \{O\}$ is generated uniformly at random from the interval $[0, 10]$; we consider $\mathbf{d} = \mathbf{1}$ (each link has the same installation cost) and an unitary bandwidth $b_{\text{wd}} = 1$.

Table 1 compares the new `Cutting-Planes` algorithm with the standard one. Columns 1-3 describe the instance, Column 4 reports the integer optimum, Columns 5-6 indicate the computing effort (iterations and CPU time in seconds) needed by the new method to reach a gap of 20% between the upper and the lower bound, Columns 7-8 provide the total computing effort spent by the new method to fully converge, and the last two columns indicate the total computing effort of the standard `Cutting-Planes`. For 13 of the 20 instances, the new method needs less CPU time than the standard `Cutting-Planes`.

id	$\|E\|$	$\|V\|$	OPT	New Meth. *Computing Effort*				Std. Meth. *Computing Effort*	
				Gap 20%		Full convergence		Full convergence	
				iters	time[s]	iters	time[s]	iters	time[s]
1	150	25	174	40	0.37	40	0.37	91	2.51
2	150	25	164	58	1.92	85	3.44	34	0.28
3	150	25	141	74	2.02	78	2.22	93	3.8
4	150	25	131	73	0.80	79	0.89	100	1.22
5	150	25	177	35	0.31	35	0.31	48	0.4
1	200	40	371	105	1.4	105	1.4	120	1.8
2	200	40	333	65	0.84	65	0.84	94	4.7
3	200	40	332	91	2.5	104	3.4	120	5.8
4	200	40	358	51	0.96	51	0.96	83	3.2
5	200	40	298	56	0.62	56	0.62	111	5.1
1	210	45	348	72	2.33	76	2.73	91	2.51
2	210	45	348	87	6.72	344	60	122	28.3
3	210	45	318	69	0.73	69	0.73	93	13.8
4	210	45	364	159	13.4	302	61.6	105	13.6
5	210	45	382	63	2.16	63	2.16	158	25.4
1	220	50	402	82	2.8	82	2.8	135	5.64
2	220	50	414	142	3.3	192	10.9	165	8.52
3	220	50	509	119	9.7	302	68.1	135	16.2
4	220	50	432	158	8.6	178	13.9	145	12.3
5	220	50	515	79	1.9	79	1.96	165	3.22

Table 1: Comparison of the new basic `Cutting-Planes` using the intersection sub-problem and the standard basic `Cutting-Planes` on small random instances (called `random-10` instances in Table 2).

Table 1 suggests that the total running time depends more on the number of iterations than on the choice between the separation and the intersection sub-problem. In both cases, the sub-problem requires solving a few LPs, but the main computational bottleneck is the iterative resolution of the master *integer* LP associated to (3.3a)-(3.3c). The more constraints are added to this master *integer* LP, the slower the ILP solver. As such, the `Cutting-Planes` algorithms become increasingly slower as the number of iterations grows. By doubling the number of iterations, the total running time is multiplied by more than 2.

The results from this section were obtained by running `C++` programs compiled with `g++` (avec l'option `-O3`) using the `Cplex` 12.6 library for `C++`. We used a mainstream `Linux` computer with a `CPU i7-5500U`.

4.2 Advanced `Cutting-Planes` and statistical comparisons

To better evaluate the the potential of the intersection sub-problem, we now use it within the advanced Benders `Cutting-Planes` algorithm from Section 2.5. Recall that this advanced algorithm calls the separation or the intersection sub-problem on a smoothed solution instead of the optimal one, *i.e.*, it uses a smoothed query point at each iteration (see more exact details in Section 2.5.2). This can reduce oscillations of the query points along the iterations, accelerating the convergence. If the cut determined using this smoothed solution separates the current optimal solution, the smoothed cut is successful (a hit). Otherwise, the smoothed cut is unsuccessful and the algorithm calls again the separation or the intersection sub-problem on the current optimal solution. In Section 2.5.3 we also presented a practical technique to (try to) avoid certain relaxed master ILPs on which the `Cplex` ILP solver can stay blocked virtually indefinitely; such master ILPs can be occasionally generated by *all* presented Benders algorithms. The above two techniques have thus been applied both for the new Benders `Cutting-Planes` based on intersections and for the standard one based on the separation sub-problem.

We will report statistics over 20 runs, providing the average, the standard deviation and the minimum value of several performance indicators. The main studied indicators are the number of iterations, the CPU time and the number of successful smoothed cuts (hits). We will also report the percentage of CPU time spent on solving (relaxed) master ILPs associated to (3.3a)-(3.3c). Reporting statistical results is slightly complicated by the fact that all our `Cutting-Planes` algorithms have *no* random component by default. Indeed, the algorithmic descriptions from Sections 2-3 do not mention any point where an algorithm has to randomly break ties. If some LP or ILP encountered along the search has multiple optimal solutions, we let `Cplex` break ties and it always returns the same optimal solution.

However, we can quite easily randomize the algorithms by simply adding random cut-set inequalities in the beginning. More exactly, we add 10 random cut-set inequalities before launching the `Cutting-Planes` algorithm, to change the way it starts, this way changing its whole evolution. These are well-known valid inequalities [5] that we implemented as follows. We first randomly split the vertex set V in two sub-sets, a subset V_s containing the source and a subset V_t containing 10 terminals (sinks). We then impose that the edges linking vertices from V_s to vertices from V_t need to have enough installed bandwidth to carry all the demands of V_t, *i.e.*, to route all the traffic from V_s (including the source) to V_t. The following is an example of a cut-set inequality, very similar to (9) from [5]: $\sum_{\substack{\{i,j\}\in E \\ i\in V_s,\ j\in V_t}} y_{ij} \geq \sum_{j\in V_t} b_i$

We use larger instances than in the previous Section 4.1. All of them have the same installation cost for each edge (*i.e.*, $\mathbf{d} = \mathbf{1}$). The first four instance classes have an unitary bandwidth $b_{\text{wd}} = 1$ and the last one has $b_{\text{wd}} = 3$. For each graph class, we choose 4 sizes, so as to generally solve the smallest one in a time of minutes and the largest one in about one hour (between 30 and 90 minutes). These are the instance graph classes:

- instances `random-10` representing random graphs with random demands in the set $\{0,\ 1,\ 2 \dots 10\}$.
- instances `random-2` representing random graphs as above but with random demands in the set $\{1,\ 2\}$.
- instances `layered-10` representing so-called layered graphs (see below) with random demands in the set $\{0,\ 1,\ 2 \dots 10\}$. We call such graphs layered in the sense that all edges $(i, j) \in E$ need to respect

$|i-j| \leq 20$. This means that a source at vertex 0 needs a path of length at least 2 to reach vertices in the interval [21..40], a path of length at least 3 to reach vertices in $[41, 60]$, 4 for the vertices in [61.80], etc. Such layered instances are more realistic in the sense very distant vertices (hosts) are not usually directly connected (by cables) in computer networks;

– instances `layered-2` representing layered graphs as above but with random demands in the set $\{1, 2\}$.

– instances `random-10-bwd3` that are identical to the first instances `random-10` except for the fact that the links have a bandwidth of $b_{\text{wd}} = 3$.

The main results of the advanced `Cutting-Planes` for both the intersection and the separation subproblem[9] are reported in Table 2, next page. The columns of this table can be divided into three groups:

Columns 1-5 describe the instance as follows: the graph class is provided in Column 1, the instance number (*id*) in Column 2, the number of edges $|E|$ (*i.e.*, the number of decision primal variables $\mathbf{y}$) in Column 3, the number of vertices $|V|$ in Column 4 and the optimal integer value in Column 5.

Columns 6-14 indicate the average computing effort of the new `Cutting-Planes` algorithm, as follows: statistics on the number of iterations (over 20 runs) in Columns 6-8, statistics on the CPU time (over 20 runs) in Columns 9-11, the average percentage of the total CPU time spent on solving (relaxed) integer-only master problems in Column 12, and finally the number of successful smoothed cuts (average and standard deviation) in Columns 13-14. All statistics indicate the average value, the standard deviation and the minimum over 20 runs.

Columns 15-23 indicate the average computing effort of the standard method, using the same format as above, more exactly: statistics on the number of iterations (over 20 runs) in Columns 15-17, statistics on the CPU time in Columns 18-20, the average percentage of the total CPU time spent on integer-only master problems in Column 21, and finally the number of successful smoothed cuts (average and standard deviation) in the last two columns.

The main conclusions that can be drawn from Table 2 are the following. The new `Cutting-Planes` method requires in general a lower number of iterations , which often leads to a CPU time speed-up between 1.2 and 2. In fact, in the best case, the new method reaches a CPU time speed-up of 3 (*i.e.*, it is 3 times faster) on the last instance of the `random-10` graphs. On the other hand, the new method is not always systematically faster in terms of CPU time. However, the only instances for which it is slightly slower are the smallest ones, for which the slow-down induced by the number of solved master *integer* LPs (equal to the number of iterations) is less important compared to other factors (*e.g.*, loading the initial program with the cut-set constraints).

Let us now examine the progress over the iterations of both Benders `Cutting-Planes` algorithms. Figures 4-6 (page 19) depict the values of the lower and the upper bounds generated by the new `Cutting-Planes` (on three instances), compared to those of the standard `Cutting-Planes` (lower bounds only). Notice that the upper bound is not available during the first (degenerate) phase of the new `Cutting-Planes`, associated to the degenerate case (i) described in Theorem 1 (Section 2.3) or in Observation 3 (Section 3.3), *i.e.*, the rays $\mathbf{0} \rightarrow \mathbf{r}$ do not even "touch" the feasible area. The initial degenerate phase is longer in Figure 4, and this is the main weak point of the new algorithm, as already stated in Section 2.5.1. However, even in Figure 4, the new upper bounds are useful to close the gap earlier at the end of the convergence (tail cutting). In Figure 6 on an instance with a bandwidth of $b_{\text{wd}} = 3$, the gap between the two bounds of the new `Cutting-Planes` algorithm is roughly 40% after only a sixth of the total number of iterations.

Figures 4-6 also suggest that the lower bounds of both methods are relatively strong, *i.e.*, they can reach about 90% of the optimum after only a fifth of the total number of iterations. During the second half of the search, the new Benders `Cutting-Planes` can generally no longer improve the lower bound substantially, but it actually tries to prove that this lower bound is close to optimal. Upper bounds can thus be very useful to reduce or close the gap, and it seems more difficult to generate quality upper bounds than quality lower bounds.

[9] The `C++` source code is publicly available on-line at `cedric.cnam.fr/~porumbed/benders/`, along with several instances.

Instance					`Cutting-Planes` using the Intersection sub-problem					`Cutting-Planes` using the Separation sub-problem				
Graph class	*id*	$\|E\|$	$\|V\|$	OPT	Iterations	Time total [secs]		Time solve	smooth cut hits	Iterations	Time total [secs]		Time solve	smooth cut hits
					avg (std) min	avg (std)	min	ILPs	avg (std)	avg (std) min	avg (std)	min	ILPs	avg (std)
`random-10` (max demand 10)	1	600	90	844	111 (7.3) 96	62.9 (17.5)	31.2	77%	44.2 (7.1)	271 (36.3) 194	119 (48.1)	44.8	84%	264 (35.4)
	2	600	90	837	111 (9.2) 96	86.5 (26.7)	43.5	83%	38 (10.1)	242 (35) 182	106 (64.9)	35.1	83%	239 (35)
	1	1000	110	932	115 (7.5) 105	157 (79.9)	36.6	86%	40.9 (8.5)	313 (68.5) 216	210 (179)	35.3	82%	306 (66.6)
	2	1000	110	984	133 (10.8) 118	218 (100)	47	85%	38.1 (10.2)	349 (56) 265	275 (252)	61.5	84%	340 (56.8)
	1	1500	130	1247	162 (15.3) 142	536 (219)	181	85%	55 (16.4)	532 (85.4) 395	1220 (1045)	163	90%	515 (84.8)
	2	1500	130	1123	153 (12.3) 138	545 (198)	235	81%	39.5 (13.6)	502 (62.9) 383	1080 (828)	186	90%	486 (61.7)
	1	2000	150	1372	184 (11.8) 162	1533 (342)	834	91%	53.8 (13.2)	804 (196) 435	5013 (2687)	796	95%	779 (191)
`random-2` (max demand 2)	1	600	90	234	168 (19.1) 138	359 (135)	125	96%	71.8 (17)	269 (28.7) 216	435 (108)	211	96%	264 (26.8)
	2	600	90	243	131 (15.9) 114	151 (56.6)	52	89%	57.7 (17.3)	274 (42.1) 206	157 (77.5)	42.7	87%	272 (41.8)
	1	1000	100	242	148 (14.5) 128	464 (144)	236	90%	51.9 (13.6)	404 (66.4) 307	757 (449)	190	91%	399 (64.6)
	2	1000	100	219	130 (11.4) 115	408 (90.9)	191	91%	53.9 (11)	396 (56.6) 316	493 (300)	110	87%	389 (57)
	1	1300	110	262	148 (10.8) 129	512 (130)	179	92%	61 (10)	562 (136) 401	1082 (675)	244	89%	546 (134)
	2	1300	110	261	148 (8) 134	546 (149)	304	88%	59.5 (8.1)	513 (85) 393	1031 (769)	226	89%	499 (84.2)
	1	1500	120	292	163 (12.6) 145	750 (224)	385	92%	70.7 (11)	577 (115) 436	1269 (1024)	243	88%	560 (116)
`layered-10` (max demand 10)	1	200	80	1556	112 (8.4) 91	31 (14.1)	13.9	93%	66.7 (10)	145 (10.6) 126	28.4 (13.4)	9.4	94%	142 (10)
	2	200	80	1766	135 (14.4) 116	27.7 (15.3)	11.4	90%	81.9 (13.6)	184 (13.6) 154	42.9 (16.1)	15.8	95%	179 (13.3)
	1	400	100	1665	171 (13.5) 153	365 (65.1)	275	96%	84.9 (13.8)	250 (14.7) 221	525 (57.2)	385	98%	247 (14.4)
	2	400	100	2166	141 (15.2) 113	221 (77.5)	53.2	96%	73.2 (15.5)	210 (19.2) 170	298 (76.5)	119	98%	207 (18.9)
	1	600	120	2873	251 (16.7) 219	1183 (95.3)	1016	97%	143 (17.5)	384 (21.4) 357	1727 (159)	1500	98%	381 (21.4)
	2	600	120	2384	221 (16.4) 190	929 (121)	711	96%	116 (16.2)	339 (26.6) 292	1369 (197)	1076	97%	335 (26.1)
	1	800	140	3136	282 (27.5) 231	1517 (212)	1149	97%	156 (26)	425 (25.7) 380	2152 (198)	1753	98%	420 (24.8)
`layered-2` (max demand 2)	1	200	80	369	125 (11.7) 101	70.7 (20.7)	27	97%	76.5 (13.9)	164 (14.9) 138	85.1 (22.1)	54.5	97%	158 (13.6)
	2	200	80	384	105 (9) 87	44.7 (21.6)	17.6	95%	56.9 (7.8)	133 (15.2) 111	45 (14.9)	20.5	95%	129 (14)
	1	400	100	503	172 (17.5) 149	373 (118)	240	98%	87.9 (15.2)	236 (14.1) 200	507 (68)	338	98%	230 (12.9)
	2	400	100	502	158 (15.1) 138	250 (47.1)	157	95%	66.8 (17.6)	232 (13.4) 210	405 (71.4)	299	98%	226 (12.7)
	1	600	120	642	197 (11.1) 179	782 (87.1)	638	99%	88.7 (10.5)	293 (24.7) 235	1149 (162)	754	98%	289 (24.3)
	2	600	120	692	200 (16.7) 168	776 (123)	523	96%	79 (15.9)	315 (20.5) 266	1249 (124)	964	98%	310 (19.6)
	1	800	140	856	277 (27.6) 221	1506	success rate: 11/20			418 (22.5) 366	2198	success rate: 11/20		
`random-10-bwd3` (bandwidth 3)	1	70	25	87	94.1 (15.1) 73	13.4 (7.6)	3.8	96%	70.9 (14.5)	116 (11.2) 90	16.8 (7.9)	5.9	97%	101 (10.2)
	2	70	25	72	124 (15.2) 107	39.1 (13.4)	20.1	98%	82.9 (10.9)	165 (28.5) 113	49.4 (25.9)	14.5	99%	133 (17.3)
	1	80	30	100	190 (37.2) 120	93.9 (35.3)	32.1	99%	146 (32.7)	276 (45.2) 189	156 (48.5)	58.9	99%	224 (30.8)
	2	80	30	88	178 (28.5) 126	83.2 (29.5)	16.5	99%	128 (23.6)	244 (39.4) 185	114 (39.3)	52.3	99%	194 (24.3)
	1	90	35	134	1124 (398) 729	1438	success rate: 13/20			1128 (111) 1024	1359	success rate: 3/20		
	2	90	35	128	710 (161) 451	739	success rate: 8/20			– – –	–	all 20/20 runs failed		
	1	100	40	192	663 (135) 402	1246	success rate: 13/20			– – –	–	all 20/20 runs failed		

Table 2: Average results of the new method compared with the standard one over hard and large instances.

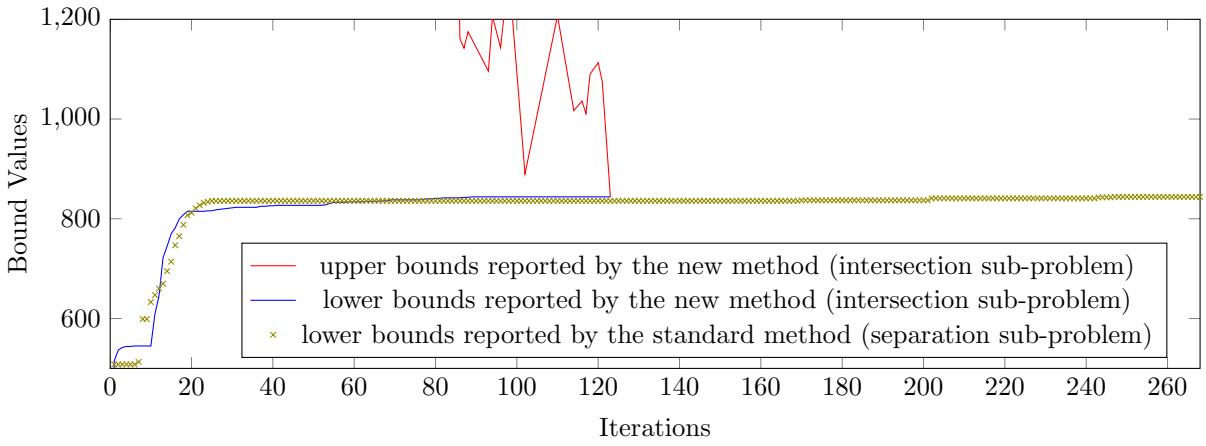


Figure 4: The running profile of the new `Cutting-Planes` method and of the standard `Cutting-Planes` method on instance $id = 1$ of graph class `random-10` with $|E| = 600$ and $|V| = 90$.

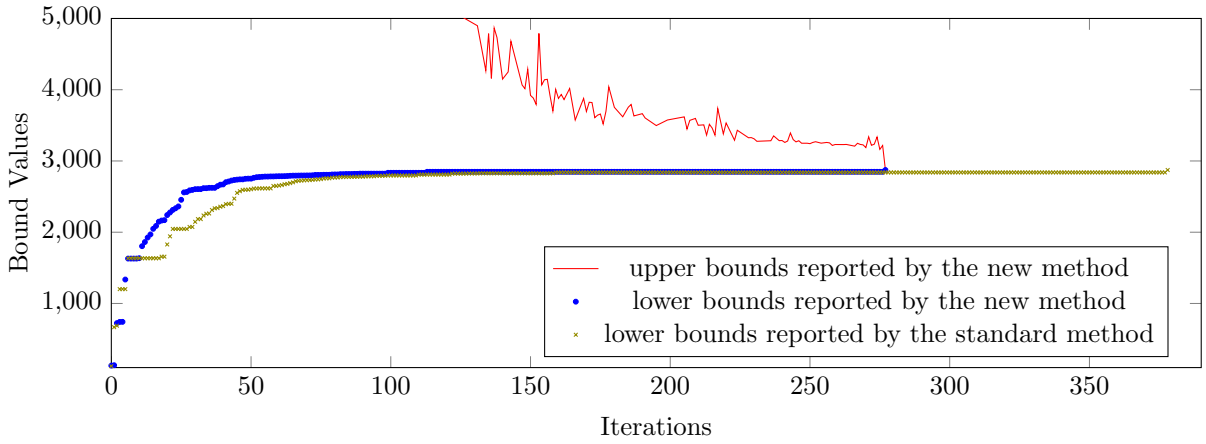


Figure 5: The running profile of the new `Cutting-Planes` method and of the standard `Cutting-Planes` method on instance $id = 1$ of graph class `layered-10` with $|E| = 600$ and $|V| = 120$.

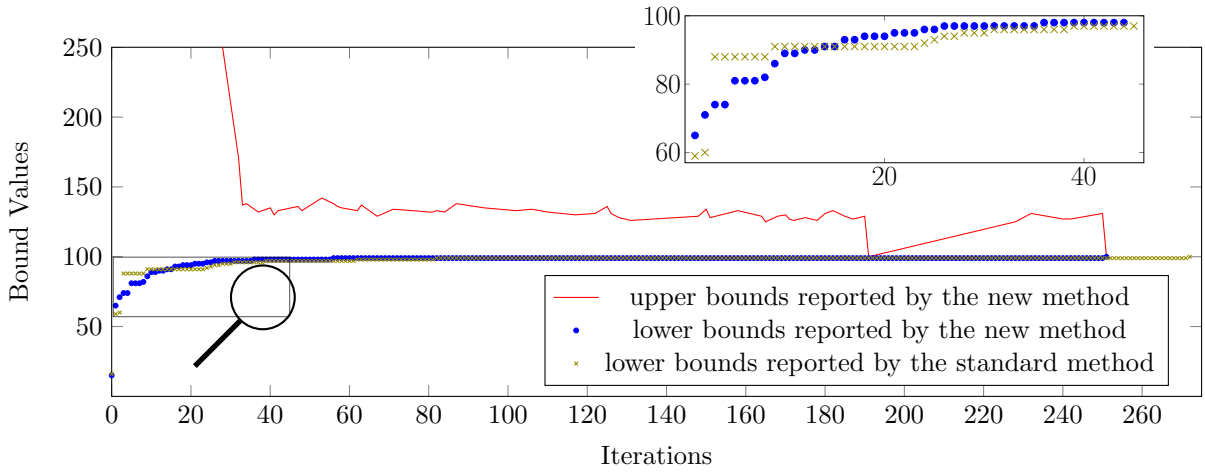


Figure 6: The running profile of the new `Cutting-Planes` and of the standard `Cutting-Planes` on instance $id = 1$ of graph class `random-10-bnd3` (hence with a bandwidth of 3) with $|E| = 80$ and $|V| = 30$.

We now investigate what are the most computationally expensive operations for both `Cutting-Planes` methods. Returning to Table 2, notice that about $80\% - 99\%$ of the total CPU time is spent on solving relaxed master ILPs (see the percents in Columns 12 and 21), for both `Cutting-Planes` methods. The total CPU time is not always exactly proportional to the number of iterations, because there are many factors that can influence the total running time. For instance, the smoothed cuts are very often sufficient for the standard `Cutting-Planes` method (next-to-last column), meaning that it is often enough to generate a unique smoothed constraint at each iteration. The new method has less successful smoothed cuts (Column 13), meaning that it often has to generate two constraints at each iteration, leading to (relaxed) master ILPs with more generated constraints, increasing the CPU time.[10]

A natural question that arises is whether the new method achieves similar or better speed-ups on larger graphs. On the first `random-10` instances of Table 2 one can simply note the following CPU time speed-ups: 1.33 for the instance $id = 1$ with $|E| = 1000$, 2.27 for the instance $id = 1$ with $|E| = 1500$ and 3.27 for the instance $id = 1$ with $|E| = 2000$. This already suggests a speed-up increase hypothesis: the speed-up obtained by the new method increases with the instance size. Table 3 below presents the speed-ups obtained on `random-10` graphs ($id = 1$) with $|E|$ ranging from 1000 to 5000. The reported figures represent averages over 5 runs, except for the first two instances with $|E| \leq 2000$ for which we simply copied the CPU time values from Table 2 (where we used 20 runs). This table confirms the above speed-up increase hypothesis, *i.e.*, the speed-up obtained by the new method becomes increasingly higher as the graph size grows, reaching speed-ups close to 10, when $|E| = 5000$. More detailed results on these new runs are publicly available on-line at `cedric.cnam.fr/~porumbed/benders/`, along with the code source.

	$\|E\|$ 1500	$\|V\|$ 130	$\|E\|$ 2000	$\|V\|$ 150	$\|E\|$ 2500	$\|V\|$ 175	$\|E\|$ 3000	$\|V\|$ 200	$\|E\|$ 3500	$\|V\|$ 225	$\|E\|$ 5000	$\|V\|$ 275
CPU Time speed-up	2.27		3.27		3.78		4.49		5.65		9.21	
CPU Time new meth.	536		1533		1886		2912		4164		8636	
CPU Time old meth.	1220		5013		7124		13063		23521		79564	
Iters new meth.	162		184		221		257		280		340	
Iters old meth.	532		804		961		1179		1518		1983	

Table 3: Speed-up of the new `Cutting-Planes` compared to the standard one on larger `random-10` graphs (all with $id = 1$). The figures for the first two graphs are copied from Table 2.

Comparing the graph classes, Table 2 suggest that the instances with lower demands (*i.e.*, `random-2` or `layered-2`) are more difficult. This is why we needed to reduce the sizes of the `random-2` instances compared to those of `random-10` in Table 2. However, we could keep the same instance sizes for the `layered-2` and the `layered-10` instances, but notice in Table 2 that some of the `layered-2` instances could not be solved by all runs, *i.e.*, certain runs remained completely blocked trying to solve a prohibitively-hard (relaxed) master ILP as described in Section 2.5.3.

However, the most difficult class of instances is by far the last one (`random-10-bwd3`) that uses a bandwidth of 3. In this case, the optimal integer solution always wastes some capacity (bandwidth), at least for any demand that is not a multiple of 3. Experiments suggest that if one adds *ten* more edges (compare results with $|E| = 80$, $|E| = 90$ and $|E| = 100$ in Table 2), both `Cutting-Planes` methods can easily require *twice* more CPU time. When setting $|E| \geq 100$, both methods have high chances of failing – notice the last two rows in which we prefer to provide the success rates instead of detailed CPU time statistics. We will

[10] Let us exemplify this on the first instance of Table 2. The standard `Cutting-Planes` requires an average of 271 iterations and the smoothed constraint is sufficient in most of them (next-to-last column indicates an average of 264); the remaining 271-264 iterations need two constraints, leading to a total of 264+2×(271-264)=276 generated constraints along all iterations. In average, the new method needed 111 iterations but only about 44 of the smoothed cuts were sufficient. This means that for about $111 - 44$ iterations in average, the new `Cutting-Planes` needed to add more than one constraint per iteration, leading to a total of roughly $44 + 2 \times (111 - 44) = 178$ generated constraints. The ratio (speed-up) of the average CPU times is $\frac{119}{62.9} \approx 1.89$, the ratio of the numbers of iterations is $\frac{271}{111} \approx 2.44$ and the ratio of the numbers of generated constraints is roughly $\frac{276}{178} \approx 1.55$. Notice the CPU time ratio is sandwiched by the ratio of the other two indicators, that can both influence the total CPU time.

thus use this more difficult instance class to solve in Section 4.3 the linear relaxation of our models (using larger instances), discussing the potential of the new upper bounds in a `Branch-and-Bound` method.

4.3 The potential of the new upper bounds to improve a `Branch-and-Bound` method based on a linear relaxation

As stated in Section 4.2 above, the instances `random-10-bwd3` are by far the most difficult. Both `Cutting-Planes` algorithms can only rarely solve such instances with $|E| > 100$ in reasonable time, while they do solve instances with thousands of edges for the other graph classes. We here consider large `random-10-bwd3` instances, taking the same sizes (with $|E|$ from 600 to 2000) used in Table 2 for the `random-10` graphs of unitary bandwidth (notice that the `random-10-bwd3` instances use the same underlying graphs as `random-10`, but only the bandwidth differ). As it stands, we do not know how to find the optimum integer solutions of such large `random-10-bnd3` instances. We can only solve their linear relaxation, in which we assume that `y` can be fractional in the initial model (3.1.a)-(3.1e), *i.e.*, allow the installation of a fractional number of transmission facilities along the edges. We can apply the same `Cutting-Planes` algorithms but instead of iteratively solving a relaxed master ILP (associated to (3.3a)-(3.3c)), we solve a linear relaxation of this master ILP, also referred to as the relaxed master *LP*.

Class	id	$\|E\|$	$\|V\|$	max demand	OPTLP	`Cutting-Planes` with intersections				Standard `Cutting-Planes`		
						UB	iters	time[s]	time solve primal LPs	iters	time[s]	time solve primal LPs
`random-10-bwd3`	1	600	90	10	281.33	311	141	16.1	1.78%	283	20.8	4.08%
	2	600	90	10	279	306	138	16.9	1.84%	243	18.6	4.14%
	1	1000	110	10	310.67	346	125	34	1.14%	277	53	2.81%
	2	1000	110	10	328	361	151	46.5	1.46%	391	70.1	4.1 %
	1	1500	130	10	415.67	461	167	99.5	1.05%	361	133	2.57%
	2	1500	130	10	374.33	421	155	88.1	0.859%	396	143	2.53%
	1	2000	150	10	457.33	507	216	215	1.14%	480	268	2.84%
	2	2000	150	10	430	482	223	228	0.723%	494	271	2.84%
`random-100-bwd3`	1	600	90	100	2918.7	2948	148	15.7	2.12%	235	17.5	3.84%
	2	600	90	100	2782	2812	109	12.8	2.03%	224	17.5	3.06%
	1	1000	110	100	3414.7	3454	749	170	3.78%	320	61.6	3.89%
	2	1000	110	100	3080.3	3117	163	48.4	1.48%	328	64.9	3.12%
	1	1500	130	100	3922	3960	895	388	2.26%	408	155	2.82%
	2	1500	130	100	4033.7	4077	955	403	2.34%	529	196	4.14%
	1	2000	150	100	4638	4688	209	199	1.11%	563	309	3.95%
	2	2000	150	100	4584	4636	1114	679	2 %	494	280	2.93%
`random-300-bwd3`	1	600	90	300	7265.7	7295	126	15	2 %	247	19.1	3.86%
	2	600	90	300	8154.7	8186	241	23	2.57%	207	15.2	3.07%
	1	1000	110	300	9060.3	9095	130	38.4	1.24%	335	65.8	3.69%
	2	1000	110	300	10163	10203	3367	733	20.1%	379	71.7	3.97%
	1	1500	130	300	12607	12652	619	297	1.73%	470	162	3.1 %
	2	1500	130	300	12003	12051	670	272	1.81%	503	176	2.14%
	1	2000	150	300	13314	13370	199	189	1.3 %	537	286	2.86%
	2	2000	150	300	13358	13408	197	182	0.934%	545	290	2.84%

Table 4: Results on the linear relaxation of the `random` instances with bandwidth $b_{\text{wd}} = 3$, using different values of the maximum demand (Column 5). The gap between the lower bound (fractional optimum in Column 6) and the integer upper bound obtained by solving intersection sub-problems (Column 7) is relatively small, varying from 10% to less than 0.5%.

Table 4 above presents the results of both `Cutting-Planes` methods on the relaxed instances with bandwidth $b_{\text{wd}} = 3$ mentioned above. The first five columns describe the instance and the maximum demand, Column 6 indicates the fractional optimum, Column 7 provides the *integer upper bound* reported

by the `Cutting-Planes` algorithm based on the intersection sub-problem. Columns 8-10 and respectively Columns 11-13 report the computing effort needed by the new and respectively the standard `Cutting-Planes` to fully converge. For each of these two algorithms, we provide three columns indicating the number of iterations, the total CPU time and the percentage of the CPU time spent on solving relaxed master *LP*s (linear relaxations of master ILPs).

The most important conclusion that can be drawn from Table 4 relates to the quality of the gab between the integer upper bound and the fractional optimum of the LP relaxation. This can vary from about 10% (instances with maximum demand 10) to less than 0.5% (instances with maximum demand 300). It is well-known that the effectiveness of a `Branch-and-bound` algorithm depends substantially on the quality of the bounds at the root node of the search tree. We can safely conclude that the `Cutting-Planes` algorithm based on the intersection sub-problem can provide strong upper bounds (leading to small gaps) at the root node, at least for instances with higher demands (see the last rows of Table 4). This suggests that the approach could be successfully embedded in a `Branch-and-bound`, but the evaluation of such an algorithm lies outside the scope of this paper.

Exceptional cases aside (*e.g.*, besides instances of high demand with $|E| = 1500$), the new `Cutting-Planes` algorithm is quite often faster than the standard one. It can reach a speed-up of up to a factor of 2 in terms of the number of iterations (see instance 2 for $|E| = 600$ or $|E| = 1000$ with a maximum demand of 100), but the CPU time speed-up is smaller than the iteration speed-up. This may be explained due to the fact that we can no longer say that the computational bottleneck of the algorithm is the resolution of the relaxed master *LP* (see the percents in Columns 10 and 13), because it is far easier to solve an LP than an ILP. Since the intersection sub-problem needs to solve more LPs than the separation sub-problem, the intersection iterations are slower than the separation iterations. For the instances mentioned above with an iteration speed-up of 2, the CPU time speed-up is about 1.5.

5 Conclusions and Prospects

We showed it is possible to improve a standard `Cutting-Planes` algorithm by "upgrading" the standard separation sub-problem to the intersection sub-problem. We focused on ILPs with prohibitively-many (feasibility) cuts in a Benders decomposition model. An advantage of the intersection sub-problem is that it allows one to calculate both a lower and an upper bound (feasible solution) at each `Cutting-Planes` iteration. We explained how solving the intersection sub-problem is equivalent to normalizing all cuts and separating (the normalized cuts). This normalization interpretation shows that the intersection sub-problem can find stronger cuts, as argued in Section 2.5.1. Under the (mild) assumptions of Theorem 1, we proved that the intersection sub-problem can be solved within the same asymptotic running time as the separation one, *i.e.*, it has the computational complexity of solving an LP over the Benders sub-problem polytope $\mathcal{P}$.

For both sub-problems, we first performed numerical tests using a relatively basic Benders `Cutting-Planes` (in Section 4.1), followed by a more advanced `Cutting-Planes` version (in Section 4.2). For instance, the advanced `Cutting-Planes` version uses solution smoothing techniques to reduce the (strong) oscillations of the current optimal solution along the iterations (see Section 2.5.2), for both sub-problems. In certain cases, the advanced version of the new `Cutting-Planes` algorithm still uses the separation sub-problem (combined with the intersection sub-problem) to finish more rapidly an initial problematic phase in which the rays do not even "touch" the feasible polytope. Once this issue solved, the new advanced `Cutting-Planes` algorithm converges more rapidly than the standard one. Experiments suggest that the new upper bounds can also be potentially useful in a `Branch-and-Bound` algorithm, because they can generate quite strong gaps (between 10% and 0.5%) at the root of the search tree.

Looking ahead, the intersection sub-problem could be potentially useful to overcome certain limitations of current practices on canonical `Cutting-Planes`, since it allows one to generate feasible solutions along the iterations. This is the second paper after [12] in a series of planned work that relate to the intersection sub-problem in different polytopes with prohibitively-many constraints, arising in various mathematical programming fields. For instance, an interesting avenue for further research could be the study of the intersection sub-problem for the primal LP models of the two problems in Appendix A (that use no Benders decomposi-

tion), for a Benders decomposition model with non-zero flow costs (Appendix B), in robust programs with prohibitively-many constraints, etc.

Acknowledgments I am grateful to two reviewers whose valuable comments helped me improve the paper.

A Prospects for applying the new method to other problems

We here only present two other problem examples that fit well the general LP (1.1) on which intersection ideas could be applied. Consider a graph $G = (V, E)$ and let us associate the decision variables $\mathbf{y} \geq \mathbf{0}$ to the edges E. We ask to minimize $\sum_{e \in E} y_e$ subject to prohibitively-many constraints of the form below, where $f : 2^E \to [0, \infty)$ is a function that can be defined as exemplified next.

$$\sum_{e \in S} y_e \geq f(S) \ \forall S \subseteq E$$

– Consider $f(S) \geq 1$ if S is a length-bounded path between a source and some destination(s) in G and $f(S) = 0$ otherwise. The resulting problem is essentially the LP (1) from [1]. The value $f(S)$ could represent a number of units (*e.g.*, surveillance cameras) that need to be placed along the path indicated by S. As an application example, consider the problem of finding the minimum number cameras needed to visualise at least once any train travelling along a length-bound path S. We can define $f(S) = 1$ if S is a length-bounded path (or $f(S) > 1$ if S has actually the capacity to accommodate multiple trains) and $f(S) = 0$ otherwise.

– Suppose one has to install facilities on certain operative (installable) edges $\widetilde{E} \subsetneq E$, considering it is impossible to install any facility on the non-operative edges $E - \widetilde{E}$, equivalent to imposing $y_{\tilde{e}} = 0 \ \forall \tilde{e} \in E - \widetilde{E}$. Define $f(S) = \gamma \cdot |S|$ if S are the edges of an induced subgraph of G of a given minimum size, or $f(S) = 0$ otherwise, for a parameter $\gamma > 0$. Any such induced S should contain at least one operative edge, but it can also contain non-operative edges. This model asks to place more facilities on denser induced subgraphs than on sparser induced subgraphs. In a road network, one can often need more facilities (*e.g.*, electric vehicle charging points, public bikes, etc.) servicing denser areas than sparser areas.

Since the coefficients of the above constraints are all non-negative, a fractional feasible solution can be converted into an integer feasible solution by rounding it up (as in Observation 4, p. 14).

B The intersection sub-problem for non-zero flow costs, for Benders reformulations with both feasibility and optimality cuts

The intersection algorithm from Theorem 1 (p. 5) can be extended to the case where the flow costs are non-zero ($\mathbf{c} \neq \mathbf{0}$). This section follows the reasoning from Theorem 1 and it presents all the modifications needed to address the case of non-zero flow costs.

Recall that Theorem 1 finds the minimum t^* such that $t^*\mathbf{r}$ is feasible with regards to the constraints (2.7c) that are associated to a zero flow cost $\hat{z} = 0$. If we consider non-zero positive flow costs, we need to check the feasibility with regards to two types of constraints (2.5b) and (2.5c), that can lead to some non-zero flow cost $\hat{z} \geq 0$. We can re-formulate (2.5b) and (2.5c) resp. as:

$$\hat{z} \geq (\mathbf{b} - \mathbf{B}\mathbf{y})^\top \mathbf{u} \ \forall \mathbf{u} \in \mathcal{P} \tag{B.1a}$$

$$0 \geq (\mathbf{b} - \mathbf{B}\mathbf{y})^\top \mathbf{u}^e \quad \text{for any extreme ray } \mathbf{u}^e \in \mathcal{P}, \tag{B.1b}$$

where $\mathcal{P}$ is the Benders polyope $\mathcal{P} = \left\{\mathbf{u} \geq \mathbf{0} : \mathbf{A}^\top \mathbf{u} \leq \mathbf{c}\right\}$, as defined in (2.4). The second above constraint set (feasibility cuts) can actually be seen as a consequence of the first constraint set (optimality cuts). If a feasibility constraint (B.1b) does not hold for some ray $\mathbf{u}^e \in \mathcal{P}$, then (B.1a) does not hold either for some $\mathbf{u} = \alpha \mathbf{u}^e$ with a large-enough $\alpha > 0$. Indeed, if $0 < (\mathbf{b} - \mathbf{B}\mathbf{y})^\top \mathbf{u}^e$, then $(\mathbf{b} - \mathbf{B}\mathbf{y})^\top \alpha \mathbf{u}^e$ can become arbitrarily

large when $\alpha \to \infty$, and so, (B.1a) can not hold for all $\mathbf{u} = \alpha \mathbf{u}^e$. This means that if all constraints (B.1a) all valid, then so are all constraints (B.1b). Thus, it is enough to focus on the optimality cuts (B.1a) and we will see they can actually be used by the intersection sub-problem algorithm to eventually generate both optimality cuts and feasibility cuts. Perhaps rather contrary to what it may seem at first glance, it has been noted [5] that restricting $\mathcal{P}$ to its extreme rays is not necessarily more efficient; this is in line with our approach of not focusing on searching extreme rays in (B.1b) but rather using (B.1a).

Let us denote the input of the intersection sub-problem by $(\mathbf{r},\ \hat{z}_r)$. Replacing $\mathbf{y}$ in (B.1a) with $t^*\mathbf{r}$ (as we did when we derived (2.8) in Theorem 1) and $\hat{z}$ with $t^*\hat{z}_r$, the intersection sub-problem requires finding the minimum $t^* \geq 0$ that satisfies the following:

$$t^* \left(\hat{z}_r + (\mathbf{Br})^\top \mathbf{u}\right) \geq \mathbf{b}^\top \mathbf{u},\ \forall \mathbf{u} \in \mathcal{P}. \tag{B.2}$$

First, we can already see that this constraint set can be seen as an extended version of (2.8). Indeed, if we extend $\mathbf{u}$ (resp. $\mathbf{b}$) to $\widehat{\mathbf{u}}$ (resp. $\widehat{\mathbf{b}}$) by adding a 0-indexed value $\widehat{u}_0 = 1$ (resp. $\widehat{b}_0 = 0$), the inequality (B.2) above can be written as $t^* \left[\hat{z}_r,\ (\mathbf{Br})^\top\right] \widehat{\mathbf{u}} \geq \widehat{\mathbf{b}}^\top \widehat{\mathbf{u}}$. This is exactly a form like (2.8) from Theorem 1, the only difference being the definition of feasible $\widehat{\mathbf{u}}$, *i.e.*, we need to say all feasible $\widehat{\mathbf{u}} = \begin{bmatrix} \widehat{u}_0 \\ \mathbf{u} \end{bmatrix}$ satisfy $\widehat{u}_0 = 1$ and $\mathbf{u} \in \mathcal{P}$. This shows we are not so far from the setting of Theorem 1.

Secondly, by strictly following Theorem 1, one can find the same cases (i), (ii.a) and (ii.b), as well as the same solution methods to address them.

(i) As in Theorem 1, this degenerate case arises when there is no $t^* \geq 0$ that satisfies (B.2). This can only happen when there is some $\mathbf{u}^\iota \in \mathcal{P}'$ such that $\hat{z}_r + (\mathbf{Br})^\top \mathbf{u}^\iota \leq 0$ and $\mathbf{b}^\top \mathbf{u}^\iota > 0$. To detect such cases, it is enough to maximize an LP very similar to (2.9) from Theorem 1, *i.e.*, it is enough to check if $0 < \max\left\{\mathbf{b}^\top \mathbf{u} :\ \mathbf{u} \in \mathcal{P},\ \hat{z}_r + (\mathbf{Br})^\top \mathbf{u} \leq 0\right\}$.

(ii.a) This case corresponds to $\hat{z}_r + (\mathbf{Br})^\top \mathbf{u} \leq 0$ and $\mathbf{b}^\top \mathbf{u} = 0$ for some $\mathbf{u} \in \mathcal{P}$, similarly to the same case in Theorem 1. The only difference is the apparition of the first term $\hat{z}_r$ indicating the flow cost. However, we can detect this case with the same approach as in Theorem 1, in particular we can take $\mathbf{u}^0 = arg\ \min\left\{\hat{z}_r + (\mathbf{Br})^\top \mathbf{u} :\ \mathbf{u} \in \mathcal{P},\ \mathbf{b}^\top \mathbf{u} = 0\right\}$ to distinguish between $\hat{z}_r + (\mathbf{Br})^\top \mathbf{u}^0 = 0$ and $\hat{z}_r + (\mathbf{Br})^\top \mathbf{u}^0 < 0$, corresponding to sub-cases (ii.a.1) and (ii.a.2). An intersection sub-problem algorithm would then take the same decisions as in Theorem 1. More exactly, if $\hat{z}_r + (\mathbf{Br})^\top \mathbf{u}^0 < 0$, the algorithm stops returning a constraint $\mathbf{u}^0$ that separates all $(t\mathbf{r}, t\hat{z}_r)$ with $t > 0$, as in (sub-)case (ii.a.1). Otherwise, the constraint defined by $\mathbf{u}^0$ allows all $t(\mathbf{r}, \hat{z}_r)$ to be feasible $\forall t \geq 0$ and we need to move to case (ii.b) to find stronger constraints.

(ii.b) This is the main (non-degenerate) case in which we need to solve the following linear-fractional program very similar to (2.10):

$$t^* = \max\left\{\frac{\mathbf{b}^\top \mathbf{u}}{\hat{z}_r + (\mathbf{Br})^\top \mathbf{u}} : \mathbf{u} \in \mathcal{P},\ \hat{z}_r + (\mathbf{Br})^\top \mathbf{u} > 0\right\}, \tag{B.3}$$

As in Theorem 1, this linear-fractional program can be solved by converting it to an LP using the Charnes-Cooper transformation [3]. More exactly, the transformation

$$\overline{\mathbf{u}} = \mathbf{u} \cdot \frac{1}{\hat{z}_r + (\mathbf{Br})^\top \mathbf{u}};\ \overline{s} = \frac{1}{\hat{z}_r + (\mathbf{Br})^\top \mathbf{u}} \tag{B.4}$$

translates the above linear-fractional program (B.3) to the equivalent pure linear program:

$$\begin{aligned} t^* = \max\ & \mathbf{b}^\top \overline{\mathbf{u}} \\ & \mathbf{A}^\top \overline{\mathbf{u}} \leq \mathbf{c}\overline{s} \\ & (\mathbf{Br})^\top \overline{\mathbf{u}} + \hat{z}_r \overline{s} = 1 \\ & \overline{\mathbf{u}} \geq \mathbf{0},\ \overline{s} \geq 0 \end{aligned}$$

If the optimum of the above (B.3) is achieved by a vertex $\mathbf{u}^o \in \mathcal{P}$, then the intersection algorithm returns the corresponding optimum t^* and an optimality cut $\hat{z}_r \geq (\mathbf{b} - \mathbf{By})^\top \mathbf{u}^o$. Otherwise, if the optimum is achieved by an extreme ray $\mathbf{u}^e$ of $\mathcal{P}$, then the algorithm returns $t^* = \lim_{\alpha \to \infty} \frac{\mathbf{b}^\top \alpha \mathbf{u}^e}{\hat{z}_r + (\mathbf{Br})^\top \alpha \mathbf{u}^e} = \frac{\mathbf{b}^\top \mathbf{u}^e}{(\mathbf{Br})^\top \mathbf{u}^e}$ and the feasibility cut $0 \geq (\mathbf{b} - \mathbf{By})^\top \mathbf{u}^o$. An extreme ray $\mathbf{u}^e$ is translated by above transformation (B.4) into $\overline{s} = 0$ and $\overline{\mathbf{u}}^e = \mathbf{u}^e \cdot \frac{1}{(\mathbf{Br})^\top \mathbf{u}^e}$, which is actually the limit point of (B.4) when $\mathbf{u}^e \to [\infty \ \infty \ \ldots \infty]$.

References

[1] G. Baier, T. Erlebach, A. Hall, E. Köhler, P. Kolman, O. Pangrác, H. Schilling, and M. Skutella. Length-bounded cuts and flows. *ACM Transactions on Algorithms*, 7(1):4:1–4:27, 2010.

[2] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische mathematik*, 4(1):238–252, 1962.

[3] A. Charnes and W. W. Cooper. Programming with linear fractional functionals. *Naval research logistics quarterly*, 9(3-4):181–186, 1962.

[4] A. M. Costa. A survey on benders decomposition applied to fixed-charge network design problems. *Computers & operations research*, 32(6):1429–1450, 2005.

[5] A. M. Costa, J.-F. Cordeau, and B. Gendron. Benders, metric and cutset inequalities for multicommodity capacitated network design. *Computational Optimization and Applications*, 42(3):371–392, 2009.

[6] C. Lee, K. Lee, and S. Park. Benders decomposition approach for the robust network design problem with flow bifurcations. *Networks*, 62(1):1–16, 2013.

[7] I. Ljubić, P. Putz, and J.-J. Salazar-González. Exact approaches to the single-source network loading problem. *Networks*, 59(1):89–106, 2012.

[8] T. L. Magnanti, P. Mirchandani, and R. Vachani. Modeling and solving the two-facility capacitated network loading problem. *Operations Research*, 43(1):142–157, 1995.

[9] T. L. Magnanti and R. T. Wong. Accelerating benders decomposition: Algorithmic enhancement and model selection criteria. *Operations research*, 29(3):464–484, 1981.

[10] J. E. Mitchell. Cutting plane methods and subgradient methods. In *Tutorials in Operations Research*, pages 34–61. (INFORMS), 2009.

[11] A. Pessoa, R. Sadykov, E. Uchoa, and F. Vanderbeck. In-out separation and column generation stabilization by dual price smoothing. In *12th International Symposium on Experimental Algorithms*, pages 354–365. Springer, 2013.

[12] D. Porumbel. Ray projection for optimizing polytopes with prohibitively many constraints in set-covering column generation. *Mathematical Programming*, 155(1):147–197, 2016.

[13] R. Rahmaniani, T. G. Crainic, M. Gendreau, and W. Rei. The benders decomposition algorithm: A literature review. *European Journal of Operational Research*, 259(3):801 – 817, 2017.

[14] V. Sridhar and J. S. Park. Benders-and-cut algorithm for fixed-charge capacitated network design problem. *European Journal of Operational Research*, 125(3):622–632, 2000.