



HAL
open science

Benders decomposition for Network Design Covering Problems

Víctor Bucarey, Bernard Fortz, Natividad González-Blanco, Martine Labbé,
Juan A Mesa

► **To cite this version:**

Víctor Bucarey, Bernard Fortz, Natividad González-Blanco, Martine Labbé, Juan A Mesa. Benders decomposition for Network Design Covering Problems. *Computers and Operations Research*, 2022. hal-03137944

HAL Id: hal-03137944

<https://inria.hal.science/hal-03137944v1>

Submitted on 10 Feb 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Benders decomposition for Network Design Covering Problems

Víctor Bucarey^a, Bernard Fortz^{c,d}, Natividad González-Blanco^b, Martine Labbé^{c,d}, Juan A. Mesa^b

^a*Data Analytics Laboratory, Vrije Universiteit Brussel, Brussels, Belgium.*

^b*Departamento de Matemática Aplicada II de la Universidad de Sevilla, Sevilla, Spain.*

^c*Département d'Informatique, Université Libre de Bruxelles, Brussels, Belgium.*

^d*Inria Lille-Nord Europe, Villeneuve d'Ascq, France.*

Abstract

We consider two covering variants of the network design problem. We are given a set of origin/destination (O/D) pairs and each such O/D pair is covered if there exists a path in the network from the origin to the destination whose length is not larger than a given threshold. In the first problem, called the maximal covering network design problem, one must determine a network that maximizes the total demand of the covered O/D pairs subject to a budget constraint on the design costs of the network. In the second problem, called the partial covering network design problem, the design cost is minimized while a lower bound is set on the total demand covered.

After presenting formulations, we develop a Benders decomposition approach to solve the problems. Further, we consider two different stabilization methods to determine the Benders cuts as well as the addition of cut-set inequalities to the master problem. Computational experiments show the efficiency of these different aspects.

Keywords: Benders decomposition, Network Design, Rapid Transit Network

1. Introduction

Infrastructure network design constitutes a major step in the planning of a transportation network since the performance, efficiency, robustness, and other features strongly depend on the selected points and the way of connecting them, see Guihare & Hao (2008). For instance, the main purpose of a rapid transit network is to improve the mobility of the inhabitants of a city or metropolitan area. This improvement could lead to lower journey times, less pollution and/or less energy consumption which drives the communities to a more sustainable mobility.

Since it is generally too expensive to connect all the existing facilities, one must determine a subnetwork that serves at best the traffic demand. Depending on the application, different optimality measures are considered. In particular, in the field of transportation, and especially in the area of

Email addresses: vbucarey@vub.be (Víctor Bucarey), bernard.fortz@ulb.ac.be (Bernard Fortz), ngonzalez2@us.es (Natividad González-Blanco), mlabbe@ulb.ac.be (Martine Labbé), jmesa@us.es (Juan A. Mesa)

passengers transportation, the aim is to get the infrastructure close to potential customers. In this framework, Schmidt & Schöbel (2014) propose to minimize the maximum routing cost for an origin-destination pair when using the new network. Alternatively, the traffic between an origin and a destination may be considered as captured if the cost or travel time when using the network is not larger than the cost or travel time of the best alternative solution (not using the new network). In this case, Perea et al. (2020) and García-Archilla et al. (2013) propose to select a sub(network) from an underlying network with the aim of capturing or covering as much traffic for a reasonable construction cost. This paper is devoted to this problem, called the **Maximum Covering Network Design Problem (MC)** as well as to the closely related problem called, **Partial Covering Network Design Problem (PC)**, in which one minimizes the network design cost for building the network under the constraint that a minimum percentage of the total traffic demand is covered.

When designing an infrastructure network, the demand is given by pairs of origin-destination points, called O/D pairs, and each such pair has an associated weight indicating the traffic between the origin and the destination. Usually this demand is encoded using an origin-destination matrix. When planning a new network, often there exists a network already functioning and offering its service to the same set of origin-destination pairs. For example, in order to improve the mobility of a big city or metropolitan area, a new rapid transit systems is planned. The current transit system could be more dense than the rapid transit planned but slower since uses the same right-of-way than the private traffic. Thus, in some way both systems compete with each other and both compete with the private mode of transportation. A similar effect occurs with mobile telecommunication operators. Therefore, the traffic between an origin and a destination is distributed among the several systems that provide the service.

There are mainly two ways of allocating the share of each mode. The first one is the binary all-or-nothing way, where the demand is covered by each mode only if the mode covers the demand point within a range of quality service, as in Church & ReVelle (1974). The second one is some continuous function, for example multi-logit probability distributions, as in Cascetta (2009). Both mode-share are based on the comparison of distances, times, costs, generalized costs or utilities. In this paper, we consider a binary one, where each O/D pair is covered only if the time spent into the network is below a threshold. This threshold represents the comparison between the time spent in the proposed network and a private mode assigning the full share to the most beneficial one.

Since most network design problems are NP-hard (see e.g. Perea et al. (2020)), recent research efforts have been oriented to apply metaheuristic algorithms to obtain good solutions in a reasonable computational time. Thus, in the field of transportation network design, Genetic Algorithms (Król & Król (2019)), Greedy Randomized Adaptive Search Procedures (García-Archilla et al. (2013)), Adaptive Large Neighborhood Searches (Canca et al. (2017)) and Matheuristics (Canca et al. (2019))

have been applied to rapid transit network design problems with applications to medium-size instances.

In this article, after presenting models for problems (MC) and (PC), we propose exact methods based on Benders decomposition. This type of decomposition has been applied to many problems in different fields, see Rahmaniani et al. (2017) for a recent literature review on the use of Benders decomposition in combinatorial optimization. One remarkable recent contribution applied to set covering and maximal covering location problems appears in Cordeau et al. (2019).

Benders decomposition for network design problems have been studied since the 80s. In Magnanti et al. (1986), authors minimize the total building cost of an uncapacitated network subject to the constraints that all O/D pairs must be covered. Given the structure of the problem, the Benders reformulation is stated with one sub-problem for each O/D pair. A Benders decomposition for a multi-layer network design problem is presented in Fortz & Poss (2009). Benders decomposition were also applied in Botton et al. (2013) in the context of designing survivable networks. Authors in Costa et al. (2009) have studied multi-commodity capacitated network design and the strength of different Benders cuts; in Marín & Jaramillo (2009) a multi-objective approach is solved through Benders decomposition where coverage is maximized, but at the same time total cost design is minimized. To best of our knowledge, this is the first time that a branch-and-Benders-cut approach is applied to network design coverage problems. We also study for the first time in this context the inclusion of *facet-defining cuts* (Conforti & Wolsey, 2019).

This paper provides several contributions. First we present new mathematical integer formulations for the network design problems (MC) and (PC). The formulation for (MC) is stronger than a previously proposed one, see e.g. Marín & Jaramillo (2009) and García-Archilla et al. (2013) (although the proposed formulation is not the main purpose of these papers), while (PC) was never studied to the best of our knowledge. Our second contribution consists of polyhedral properties that are useful from the algorithmic point of view. A third contribution is the study of exact algorithms for the network design based on different Benders implementations. We propose a normalization technique and we study the facet-define cuts. Our computational experiments show that our Benders implementations are competitive with exact and non-exact methods in the literature.

All our computational experiments were performed on a computer equipped with a Intel Core i5-7300 CPU processor, with 2.50 gigahertz 4-core, and 16 gigabytes of RAM memory. The operating system is 64-bit Windows 10. Codes were implemented in Python 3.8. These experiments have been carried out through CPLEX 12.10 solver, named CPLEX, using its Python interface. CPLEX parameters were set to their default values and the models were optimized in a single threaded mode. We used their `LazyConstraintCallback` function to separate integer solution and the `UserCutCallback` to separate fractional ones.

In tables reporting these results, \mathbf{t} denotes the average values for solution times given in seconds,

gap denotes the relative optimality gap in percent, **LP gap** denotes the LP gap in percent and **cuts** is the number of cuts generated.

The structure of the paper is as follows. In Section 2, we state the mixed integer formulation for (MC) and (PC). We also study some polyhedral properties of the formulation and propose a simple algorithm to find an initial feasible solution for both problems. In Section 3, we study different Benders implementations and some algorithmic enhancements. Also we discuss some improvements based on cut-set inequalities. A computational study is detailed in Section 4. Finally, our conclusions are presented in Section 5.

2. Problem formulations and some properties

In this section we present mixed integer formulations for the Maximal Covering Network Design Problem (MC) and the Partial Set Covering Network Design Problem (PC). We also describe some pre-processing methods and finish with some polyhedral properties. We first introduce some notation.

We consider an undirected graph denoted by $\mathcal{N} = (N, E)$, where N and E are the sets of potential nodes and edges that can be built. Each element $e \in E$ is denoted by $\{i, j\}$, with $i, j \in N$. We use the notation $i \in e$ if node i is a terminal node of e .

The mobility patterns in the metropolitan area are represented by a set $W \subset N \times N$ of origin/destination pairs named as O/D pairs. Each $w = (w^s, w^t) \in W$ is defined by an origin node $w^s \in N$, a destination node $w^t \in N$, an associated demand $g^w > 0$ and a private utility $u^w > 0$. This parameter translates the fact that there already exists a different mode of transportation, referenced as private mode, competing with the network to be built in an all or nothing way. In other words, an O/D pair (w^s, w^t) will travel on the newly built network if it contains a path between w^s and w^t of length shorter than the private utility u^w . We then say that the O/D pair is covered.

Costs for building nodes, $i \in N$, and edges, $e \in E$, are denoted by b_i and c_e , respectively. The total building cost cannot exceed the budget C_{max} .

For each $e = \{i, j\} \in E$, we define two arcs: $a = (i, j)$ and $\hat{a} = (j, i)$. The resulting set of arcs is denoted by A . The length of arc $a \in A$ is denoted by d_a . For each O/D pair $w \in W$ we define a subgraph $\mathcal{N}^w = (N^w, E^w)$ containing all feasible nodes and edges for w , i.e. that belong to a path in \mathcal{N} whose total length is lower than or equal to u^w . We also denote A^w as the set of feasible arcs. In Section 2.2, we describe how to construct these subgraphs. We use notation $\delta_w^+(i)$ ($\delta_w^-(i)$ respectively) to denote the set of arcs going out (in respectively) of node $i \in N^w$. In particular, $\delta_w^-(w^s) = \emptyset$ and $\delta_w^+(w^t) = \emptyset$. We also denote by $\delta_w(i)$ the set of edges incident to node i in graph \mathcal{N}^w .

2.1. Mixed Integer Formulations

We first present a formulation of the *Maximal Covering Network Design Problem* (MC), whose aim is to design an infrastructure network maximizing the total demand covered subject to a budget constraint:

$$(MC) \quad \max_{\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{f}} \sum_{w \in W} g^w z^w \quad (2.1)$$

$$\text{s.t.} \quad \sum_{e \in E} c_e x_e + \sum_{i \in N} b_i y_i \leq C_{max}, \quad (2.2)$$

$$x_e \leq y_i, \quad e \in E, i \in e, \quad (2.3)$$

$$\sum_{a \in \delta_w^+(i)} f_a^w - \sum_{a \in \delta_w^-(i)} f_a^w = \begin{cases} z^w, & \text{if } i = w^s, \\ -z^w, & \text{if } i = w^t, \\ 0, & \text{otherwise,} \end{cases} \quad w \in W, i \in N^w, \quad (2.4)$$

$$f_a^w + f_{\hat{a}}^w \leq x_e, \quad w \in W, e = \{i, j\} \in E^w : a = (i, j), \hat{a} = (j, i), \quad (2.5)$$

$$\sum_{a \in A^w} d_a f_a^w \leq u^w z^w, \quad w \in W, \quad (2.6)$$

$$y_i, x_e, z^w \in \{0, 1\}, \quad i \in N, e \in E^w, w \in W, \quad (2.7)$$

$$f_a^w \in \{0, 1\}, \quad a \in A^w, w \in W, \quad (2.8)$$

where y_i and x_e represent the binary design decision of building node i and edge e , respectively. Mode choice variables z^w take value 1 if the O/D pair w is covered and 0 otherwise. Variables f_a^w are used to model a path between w^s and w^t , if possible. Variable f_a^w takes value 1 if arc a belongs to the path from w^s to w^t , and 0 otherwise. For each f_a^w such that $a \notin A^w$, this variable is set to zero.

The objective function (2.1) maximizes the demand covered. Constraint (2.2) limits the total building cost. For each pair w , expressions (2.4), (2.5) and (2.6) guarantee demand conservation and link flow variables f_a^w with decision variables z^w and design variables x_e . Constraints (2.6) upper bound the length of the path for each pair w . Variable z^w will take value 1 only if there exists a path between w^s and w^t with length shorter than u^w . This path is represented by variables f_a^w . Finally, constraints (2.7) and (2.8) state that variables are binary.

The *Partial Covering Network Design Problem* (PC), which minimizes the total building cost of the network subject to a minimum coverage level of the total demand, can be formulated as follows:

$$(PC) \quad \min_{\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{f}} \sum_{i \in N} b_i y_i + \sum_{e \in E} c_e x_e \quad (2.9)$$

$$\text{s.t.} \quad \sum_{w \in W} g^w z^w \geq \beta Z_{total} \quad (2.10)$$

Constraints (2.3), (2.4), (2.5), (2.6), (2.7), (2.8)

where $\beta \in (0, 1]$ and $Z_{total} = \sum_{w \in W} g^w$. Here, the objective function (2.9) minimizes the design cost. Constraint (2.10) imposes that a proportion β of the total demand is covered.

In previous works by Marín & Jaramillo (2009) and García-Archilla et al. (2013), constraints (2.5) and (2.6) are written in a different way. For example, in García-Archilla et al. (2013), these constraints were written as

$$f_a^w + z^w - 1 \leq x_a, \quad w \in W, e = \{i, j\} \in E^w : a = (i, j), \quad (2.11)$$

$$\sum_{a \in A^w} d_a f_a^w + M(z^w - 1) \leq u^w z^w, \quad w \in W, \quad (2.12)$$

where the design variable x_a is defined per arc. Given that $z^w - 1 \leq 0$, expressions (2.5) and (2.6) are stronger than (2.11) and (2.12), respectively.

In addition, constraint (2.12) involves a “*big-M*” constant. Our proposed formulation does not need it, which avoids the numerical instability generated by this constant. In Table 1, we compare both formulations for (MC). We use CPLEX to solve some instances described in Subsection 4.1. Average solution times in seconds and percent LP gaps are shown for instances with 10 and 20 nodes. We also tested instances with 40 nodes but most of them were not solved to optimality within one hour. In that case, we provide the optimality gap instead of the solution time. We consider 5 instances of each size. Note that constraints (2.12) are equivalent to constraints (2.6) by setting $M = 0$. We tested several positive values for M . We observed that our proposed formulation is not only stronger than the one proposed in García-Archilla et al. (2013), but it is also more computationally efficient.

Network	Formulation using (2.5)-(2.6)		Formulation using (2.11)-(2.12)	
	t	LP gap	t	LP gap
N10	0.17	43.21	0.26	96.43
N20	5.78	56.33	228.22	106.71
	gap	LP gap	gap	LP gap
N40	11.74	68.15	54.85	137.13

Table 1: Comparing the performance of the two different types of mode choice and capacity constraints for (MC) within a time limit of 1 hour. The majority of N40 instances were not solved to optimality, then the average gap is shown.

2.2. Preprocessing methods

In this section we describe some methods to clean up instances before solving. First, we describe how to build each subgraph $\mathcal{N}^w = (N^w, E^w)$. Then for each problem, (MC) and (PC), we sketch a method to eliminate O/D pairs which will never be covered.

To create \mathcal{N}^w we only consider useful nodes and edges from \mathcal{N} . For each O/D pair w , we eliminate all the nodes $i \in N^w$ that do not belong to any path from w^s to w^t shorter than u^w . Then, we define

E^w as the set of edges in E incident to the non eliminated nodes. Finally, the set A^w is defined with the duplicated directed version of edges in E^w with the exception of arcs in the form (i, w^s) and (w^t, i) . We describe this procedure in Algorithm 1.

We assume that there are no nodes or edges which cannot be built because their construction cost is higher than the budget.

Algorithm 1 Preprocessing I

```

for  $w \in W$  do
     $N^w = N$ 
    for  $i \in N$  do
        compute the shortest path for the O/D pairs  $(w^s, i)$  and  $(i, w^t)$ 
        if the sum of the lengths of these paths is greater than  $u^w$  then
             $N^w = N^w \setminus \{i\}$ 
             $E^w = E^w \setminus \delta(i)$ 
        end if
    end for
     $A^w = \{(i, j) \in A : \{i, j\} \in E^w, j \neq w^s, i \neq w^t\}$ 
end for
return  $\{\mathcal{N}^w = (N^w, E^w), A^w\}_{w \in W}$ 

```

Next, we focus on (MC). We can eliminate an O/D pair w that is too expensive to be covered. That means, the O/D pair w is deleted from W if there is no path between w^s and w^t satisfying: i. its building cost is less than C_{max} ; or ii. its length is less than u^w .

This can be checked by solving a shortest path problem with resource constraints that can be done in a pseudo-polynomial time. Desrochers (1988) shows how to adapt Bellman-Ford algorithm to solve it. However, for the size of graphs that we are considering, we solve it as a feasibility problem. For each w , we consider the feasibility problem associated to constraints (2.2) (2.3), (2.4), (2.5), (2.6) and (2.7), with z^w fixed to 1. If this problem is infeasible, then the O/D pair w is deleted from W . Otherwise, there exists a feasible path denoted by Path_w . We denote by $(\tilde{E}^w, \tilde{N}^w)$ the subgraph induced by Path_w .

2.3. Polyhedral properties

Both formulations (MC) and (PC) involve flow variables f_a^w whose number can be huge when the number of O/D pairs is large. To circumvent this drawback we use a Benders decomposition approach for solving (MC) and (PC).

In this subsection, we present properties of the two formulations that allow us to apply such

a decomposition in an efficient way. The first proposition shows that we can relax the integrality constraints on the flow variables f_a^w .

Let (MC-R) and (PC-R) denote the formulations (MC) and (PC) in which constraints (2.8) are replaced by nonnegativity constraints, i.e.

$$f_a^w \geq 0, w \in W, a \in A. \quad (2.13)$$

We denote the set of feasible points to a formulation F by $\mathcal{F}(F)$. Further, let Q be a set of points $(x, z) \in R^q \times R^p$. Then the projection of Q onto the x-space, denoted $Proj_x Q$, is the set of points given by $Proj_x Q = \{x \in R^q : (x, z) \in Q \text{ for some } z \in R^p\}$.

Proposition 1. $Proj_{x,y,z} \mathcal{F}(MC) = Proj_{x,y,z} \mathcal{F}(MC - R)$ and $Proj_{x,y,z} \mathcal{F}(PC) = Proj_{x,y,z} \mathcal{F}(PC - R)$.

Proof. We provide the proof for (MC), the other one being identical.

First, $\mathcal{F}(MC) \subseteq \mathcal{F}(MC - R)$ implies $Proj_{x,y,z} \mathcal{F}(MC) \subseteq Proj_{x,y,z} \mathcal{F}(MC - R)$.

Second, let (x, y, z) be a point belonging to $Proj_{x,y,z} \mathcal{F}(MC - R)$. For every O/D pair $w \in W$ such that $z^w = 0$, $f^w = 0$. In the case where $z^w = 1$, there exists a flow $f_a^w \geq 0$ satisfying (2.4) and (2.5) that can be decomposed into a convex combination of flows on paths from w^s to w^t and cycles. Given that the flow f_a^w also satisfies (2.6), then a flow of value 1 on one of the paths in the convex combination must satisfy this constraint. Hence by taking f_a^w equal to 1 for the arcs belonging to this path and to 0 otherwise, we show that (x, y, z) also belongs to $Proj_{x,y,z} \mathcal{F}(MC)$. \square

Based on Proposition 1, we propose a *Benders decomposition* where variables f_a^w are projected out from the model and replaced by *Benders feasibility cuts*. As we will see in Section 3.3, we also consider the Benders *facet-defining cuts* proposed in Conforti & Wolsey (2019). To apply this technique it is necessary to get an interior point of the convex hull of $Proj_{x,y,z} \mathcal{F}(MC - R)$ (resp. $Proj_{x,y,z} \mathcal{F}(PC - R)$). The following property give us an algorithmic tool to apply this technique to (MC).

Proposition 2. *After pre-processing, the convex hull of $Proj_{x,y,z} \mathcal{F}(MC - R)$ is full-dimensional.*

Proof. To prove the result, we exhibit $|N| + |E| + |W| + 1$ affinely independent feasible points:

- The 0 vector is feasible.
- For each $i \in N$, the points:

$$y_i = 1, y_{i'} = 0, i' \in N \setminus \{i\}, \quad x_e = 0, e \in E, \quad z^w = 0, w \in W$$

- For each $e = \{i, j\} \in E$, the points:

$$y_k = 1, k \in e, y_k = 0, k \in N \setminus \{i, j\}, \quad x_e = 1, x_{e'} = 0, e' \in E \setminus \{e\}, \quad z^w = 0, w \in W$$

- For each $w \in W$, the points:

$$y_i = 1, i \in \tilde{N}^w, y_i = 0, i \in N \setminus \tilde{N}^w, \quad x_e = 1, e \in \tilde{E}^w, x_e = 0, e \in E \setminus \tilde{E}^w, \\ z^w = 1, z^{w'} = 0, w' \in W \setminus \{w\}$$

Clearly these points are feasible and affinely independent. Thus the polytope is full-dimensional. \square

The proof of Proposition 2 gives us a way to compute an interior point of the convex hull of $Proj_{x,y,z}(\mathcal{F}(MC - R))$. The average of these $|N| + |E| + |W| + 1$ points is indeed such an interior point.

This is not the case for (PC) as we show in Example 1.

Example 1. Consider the instance of (PC) given by the data presented in Table 2 and Figure 1. We take the case where half of the population at least must be covered, that is $\beta = 0.5$. In order to satisfy the trip coverage constraint (2.10), the O/D pair $W = (1, 4)$ must be covered. That is $z^{(1,4)} = 1$ is an implicit equality. Furthermore, the only path with a length is less than or equal to $u^{(1,4)} = 15$ is composed of edges $\{1,2\}$ and $\{2,4\}$. Hence, $x_{\{1,2\}}$, $x_{\{2,4\}}$, y_1 , y_2 and y_4 must take value 1. In consequence, the polytope associated to (PC) is not full-dimensional.

Origin	Destination	u^w	g^w
1	4	15	200
2	4	10	50
3	4	15	50

Table 2: Data in Example 1. We consider $\beta = 0.5$.

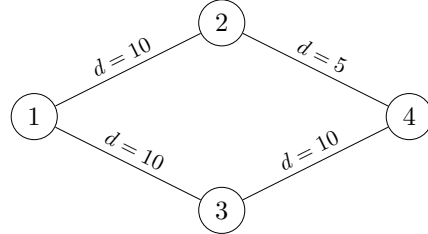


Figure 1: Graph of Example 1.

We can compute the dimension of the convex hull of $Proj_{x,y,z}\mathcal{F}(PC - R)$ in an algorithmic fashion.

We find feasible affinely independent points and at the same time we detect O/D pairs which must be covered in any feasible solution. Due to the latter, there are a subset of nodes and a subset of edges that have to be built in any feasible solution. This means that there is a subset of design variables $y_i, i \in N$, $x_e, e \in E$ and mode choice variables $z^w, w \in W$ that must take value 1. At the opposite to (MC), a solution to (PC) with all variables set to 0 is not feasible. However, the solution obtained by serving all O/D pairs and building all nodes and edges is feasible. Therefore, we start with a solution with all variables in $\mathbf{x}, \mathbf{y}, \mathbf{z}$ set to 1 and we check, one by one, if it is feasible to set them to 0. By

setting one variable x_e or y_i to 0, it may become impossible to cover some O/D pair w . In this case, we say that edge e and node i is *essential* for w . To simplify the notation, we introduce the binary parameters θ_e^w and θ_i^w taking value 1 if edge e (respectively node i) is essential for w . These new points are stored in a set L . Each time the algorithm finds a variable that cannot be set to 0, we store it in sets \bar{N} , \bar{E} , \bar{W} , respectively. At the end of the algorithm, the dimension of the convex hull of $Proj_{x,y,z}\mathcal{F}(PC - R)$ is

$$\dim(\mathcal{P}_{\mathbf{x},\mathbf{y},\mathbf{z}}) = |N| + |E| + |W| - (|\bar{N}| + |\bar{E}| + |\bar{W}|).$$

This procedure is depicted in Algorithm 2.

Algorithm 2 allows : i) to set some binary variables equal to 1, decreasing the problem size; and ii) to compute a relative interior point of the convex hull of $Proj_{x,y,z}\mathcal{F}(PC - R)$, necessary for the *facet-defining cuts*, as explained below in Section 3.3. The relative interior point is given by the average of the points in set L .

Example 1 cont. Regarding the previous example and following Algorithm 2, the O/D pair (1, 4) must be covered, $z^{(1,4)} = 1$. Due to that, as its shortest path in the networks $(N^{(1,4)}, E^{(1,4)} \setminus \{\{1, 2\}\})$ and $(N^{(1,4)}, E^{(1,4)} \setminus \{\{2, 4\}\})$ is greater than $u^{(1,4)} = 15$, variables $x_{\{1,2\}}$, $x_{\{2,4\}}$, y_1 , y_2 , y_4 are set to 1. Finally, the dimension of this polyhedron is

$$\dim(P_{\mathbf{x},\mathbf{y},\mathbf{z}}) = 4 + 4 + 3 - (3 + 2 + 1) = 5.$$

The relative interior point computed is:

$$\begin{aligned} x_{\{1,2\}} = 1, \quad x_{\{2,4\}} = 1, \quad x_{\{1,3\}} = \frac{5}{6}, \quad x_{\{3,4\}} = \frac{2}{3}, \quad y_1 = 1, \quad y_2 = 1, \quad y_3 = \frac{5}{6}, \quad y_4 = \frac{5}{6}, \\ z^{(1,4)} = 1, \quad z^{(2,4)} = \frac{5}{6}, \quad z^{(3,4)} = \frac{1}{2}. \end{aligned}$$

2.4. Setting an initial solution

We determine an initial feasible solution for (MC) and (PC) with a simple greedy heuristic in which we sequentially select O/D pairs with best ratio demand over building cost. More precisely, given the potential network $\mathcal{N} = (N, E)$, we compute for each O/D pair w the ratio $r_w = \frac{g^w}{C(\text{Path}_w)}$, where $C(\text{Path}_w)$ is the cost of a feasible path for w . We order these ratios decreasingly. We use this initial order in the heuristic both for (MC) and (PC). For (MC) the method proceeds as follows. It starts with an empty list of nodes and edges built, an empty list of O/D pairs covered, and a total cost set to 0. For each O/D pair w , in decreasing order of r_w , the heuristic tries to build Path_w considering edges and nodes that are already built. If the additional cost plus the current cost is less than the budget C_{max} , nodes and edges in Path_w are built and the O/D pair w is covered (i.e. $z^w = 1$). The

Algorithm 2 Computing the dimension of the polytope of (PC)

Initialization: Set $\bar{N} = \emptyset$, $\bar{E} = \emptyset$, $\bar{W} = \emptyset$ and $L = \emptyset$

Add to set L : $(y_i = 1, i \in N, \quad x_e = 1, e \in E, \quad z^w = 1, w \in W)$.

for $w' \in W$ **do**

if $\sum_{w \in W \setminus \{w'\}} g^w \geq \beta Z_{total}$ **then**

Add to set L : $(y_i = 1, i \in N, \quad x_e = 1, e \in E, \quad z^{w'} = 0, z^w = 1, w \in W \setminus \{w'\})$.

else

$\bar{W} = \bar{W} \cup \{w'\}$

for $e = \{i, j\} \in E$ **do**

Compute shortest path between w'^s and w'^t in the graph $(N^{w'}, E^{w'} \setminus \{e\})$.

if the length of the shortest path is greater than $u^{w'}$ **or** There is no path between w'^s and w'^t **then**

$\bar{E} = \bar{E} \cup \{e\}$ and $\bar{N} = \bar{N} \cup \{i, j\}$

end if

end for

end if

end for

for $e' \in E \setminus \bar{E}$ **do**

Add to set L : $(y_i = 1, i \in N, \quad x_e = 1, e \in E \setminus \{e'\}, x_{e'} = 0, \quad z^w = 1 - \theta_{we'}, w \in W)$.

end for

for $i' \in N \setminus \bar{N}$ **do**

Add to set L :

$(y_{i'} = 0, y_i = 1, i \in N \setminus \{i'\}, \quad x_e = 0, i' \in e, x_e = 1, i' \notin e, \quad z^w = 1 - \theta_{wi'}, w \in W)$.

end for

$\dim(\mathcal{P}_{\mathbf{x}, \mathbf{y}, \mathbf{z}}) = |N| + |E| + |W| - (|\bar{N}| + |\bar{E}| + |\bar{W}|)$.

return \bar{N} , \bar{E} , \bar{W} , L and $\dim(\text{conv}(\mathcal{P}_{\mathbf{x}, \mathbf{y}, \mathbf{z}}))$.

total cost, the lists of nodes and edges built are updated. Otherwise we proceed with the next O/D pair. At the end of the algorithm we have an initial feasible solution.

To get an initial solution for (PC) we start with a list of all the O/D pairs covered and the amount of population covered equal to Z_{total} . For each O/D pair w , in decreasing order of r_w , the algorithm checks if by deleting the O/D pair w from the list, the coverage constraint (2.10) is satisfied. If so, the O/D pair w is deleted from the list and the amount of population covered is updated. Finally, the algorithm builds the union of the subgraphs $(\tilde{N}^w, \tilde{E}^w)$ induced by Path_w for all the O/D pairs covered.

Pseudo-codes for both routines are provided in Appendix A. In Section 4, we will show the efficiency of adding this initial solution at the beginning of the branch-and-benders-cut procedure.

3. Benders Implementations

In the following, we describe different Benders implementations for (MC) and (PC) obtained by projecting out variables f_a^w . These implementations are used as sub-routines in a branch-and-Benders-cut scheme. This scheme cuts infeasible solutions along the branch-and-bound tree. Depending on the implementation, solutions can be cut whenever an integer solution is found or at any node in the tree. In the case of (MC), the master problem that we solve is:

$$\begin{aligned}
 \text{(M - MC)} \quad & \max_{\mathbf{x}, \mathbf{y}, \mathbf{z}} \sum_{w \in W} g^w z^w & (3.1) \\
 \text{s.t.} \quad & (2.2), \quad (2.3), \quad (2.7) \\
 & + \{\text{Benders Cuts } (\mathbf{x}, \mathbf{y}, \mathbf{z})\}
 \end{aligned}$$

The master problem for (PC), named (M - PC), is stated analogously.

In Section 3.1, we discuss the standard *Benders cuts* obtained by dualizing the respective feasibility subproblem. Then, in Section 3.2 we propose a way of generating normalized subproblems, we named them *normalized Benders cuts*. In Section 3.3, we apply *facet-defining cuts* in order to get stronger cuts, as it is proposed in Conforti & Wolsey (2019). Finally, we discuss an implementation where at the beginning *cut-set inequalities* are added to enhance the link between \mathbf{z} and \mathbf{x} , and then *Benders cuts* are added.

3.1. LP feasibility cuts

Since the structure of the model allows it, we consider a feasibility subproblem made of constraints (2.4), (2.5), (2.6) and (2.13) for each commodity $w \in W$, denoted by $(\text{SP})^w$. As it is clear from the context, we remove the index w from the notation. The dual of each feasibility subproblem can be

expressed as:

$$(\text{DSP})^w \quad \max_{\boldsymbol{\alpha}, \boldsymbol{\sigma}, \boldsymbol{v}} \quad z \alpha_{w^s} - \sum_{e \in E} x_e \sigma_e - u z v \quad (3.2)$$

$$\text{s.t.} \quad \alpha_i - \alpha_j - \sigma_e - d_a v \leq 0, \quad a = (i, j) \in A : e = \{i, j\} \quad (3.3)$$

$$\sigma_e, v \geq 0, \quad e \in E \quad (3.4)$$

where vector $\boldsymbol{\alpha}$ is related to constraints (2.4), $\boldsymbol{\sigma}$ is the dual variable vector corresponding to the set of constraints (2.5) and \boldsymbol{v} is the dual variable of constraint (2.6). Since one of constraints in (2.4) is linearly dependent, we set $\alpha_{w^t} = 0$. Given a solution of the master problem $(\mathbf{x}, \mathbf{y}, \mathbf{z})$, there are two possible outcomes for $(\text{SP})^w$:

1. $(\text{SP})^w$ is infeasible and $(\text{DSP})^w$ is unbounded. Then, there exists an increasing direction $(\boldsymbol{\alpha}, \boldsymbol{\sigma}, \boldsymbol{v})$ with positive cost. In this case, the solution $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ is cut by:

$$(\alpha_{w^s} - u v) z - \sum_{e \in E} \sigma_e x_e \leq 0 \quad (3.5)$$

2. $(\text{SP})^w$ is feasible and consequently, $(\text{DSP})^w$ has an optimal objective value equal to zero. In this case, no cut is added.

3.2. Normalized Benders cuts

The overall branch-and-Benders-cut performance heavily relies on how the cuts are implemented. It is known that feasibility cuts may have poor performance due to the lack of ability of selecting a *good* extreme ray (see for example Fischetti et al. (2010); Ljubić et al. (2012)). However, normalization techniques are known to be efficient to overcome this drawback Magnanti & Wong (1981); Balas & Perregaard (2002, 2003). The main idea is to transform extreme rays in extreme points of a suitable polytope. In this section we study three ways to normalize the dual subproblem described above.

First, we note that the feasibility subproblem can be reformulated as a min cost flow problem in \mathcal{N}^w with capacities \mathbf{x} and arc costs d_a .

$$(\text{NSP})^w \quad \min_{\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{f}} \sum_{a \in A} d_a f_a \quad (3.6)$$

$$\text{s.t.} \quad (2.4), (2.5), (2.13)$$

The associated dual subproblem is:

$$(\text{DNSP})^w \quad \max_{\boldsymbol{\alpha}, \boldsymbol{\sigma}} \quad z \alpha_{w^s} - \sum_{e \in E} \sigma_e x_e \quad (3.7)$$

$$\text{s.t.} \quad \alpha_i - \alpha_j - \sigma_e \leq d_a, \quad a = (i, j) \in A : e = \{i, j\} \quad (3.8)$$

$$\sigma_e \geq 0, \quad e \in E \quad (3.9)$$

Whenever $z > 0$, the primal subproblem $(\text{NSP})^w$ may be infeasible. Subproblems $(\text{NSP})^w$ are no longer feasibility problems, although some of their respective dual forms can be unbounded. As splitting demand constraint has to be satisfied there are two kind of cuts to add:

1. $(\text{NSP})^w$ is infeasible and $(\text{DNSP})^w$ is unbounded. In this case, the solution $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ is cut by the constraint:

$$\alpha_{w^s} z - \sum_{e \in E} \sigma_e x_e \leq 0 \quad (3.10)$$

2. $(\text{NSP})^w$ is feasible and $(\text{DNSP})^w$ has optimal solution. Consequently, if their solutions $(\boldsymbol{\alpha}, \boldsymbol{\sigma})$ and $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ satisfy that $\alpha_{w^s} z - \sum_{e \in E} \sigma_e x_e > u z$ then, the following cut is added

$$(\alpha_{w^s} - u) z - \sum_{e \in E} \sigma_e x_e \leq 0. \quad (3.11)$$

We refer to this implementation `BD_Norm1`.

In this situation, there still exists dual subproblems $(\text{DNSP})^w$ with extreme rays. We refer to `BD_Norm2` as second dual normalization obtained by adding the dual constraint $\alpha_{w^s} = u + 1$. In this case, every extreme ray of $(\text{SP})^w$ correspond to one of the extreme points of $(\text{NSP})^w$. A cut is added whenever the optimal dual objective function is positive. This cut is in the form:

$$z - \sum_{e \in E} \sigma_e x_e \leq 0 \quad (3.12)$$

We finally tested a third dual normalization, `BD_Norm3`, by adding constraints

$$\sigma_e \leq 1, \quad e \in E, \quad (3.13)$$

directly in $(\text{DSP})^w$.

Network	BD_Norm1		BD_Norm2		BD_Norm3	
	t	cuts	t	cuts	t	cuts
N10	0.21	44	0.22	47	0.24	104
N20	2.83	362	5.76	595	5.22	1418
N40	687.88	2904	*	*	*	*

Table 3: Comparing the performance of the three dual normalization within a time limit of 1 hour for (MC). N10, N20 and N40 are referred to networks with 10, 20 and 40 nodes respectively. In results marked with '*', four over five instances were not solved within 1 hour.

We tested the three dual normalizations described above for (MC) using randomly generated networks with 10, 20 and 40 nodes, as described in Subsection 4.1. Table 3 shows average values obtained for solution time in seconds and number of cuts needed for this experiment. The only one

that seems competitive is `BD_Norm1`. We observed that cut coefficients generated with `BD_Norm1` are mainly 0's or 1's. In the case of `BD_Norm2` and `BD_Norm3` we observe that coefficients generated are larger than the ones generated by `BD_Norm1`, so they may induce numerical instability. This situation is similar for the case of (PC).

3.3. Facet-defining Benders cuts

Here we describe how to generate Benders cuts for (MC) based on the ideas exposed in Conforti & Wolsey (2019). The procedure for (PC) is the same. Given an *interior point* or *core point*, named as $(\mathbf{x}^{in}, \mathbf{y}^{in}, \mathbf{z}^{in})$, of the convex hull of feasible solutions and a solution of the LP relaxation of the current restricted master problem, an *exterior point*, named $(\mathbf{x}^{out}, \mathbf{y}^{out}, \mathbf{z}^{out})$, a cut that defines a facet or an improper face of the polyhedron defined by the LP relaxation of $Proj_{x,y,z}\mathcal{F}(MC)$ is generated. We denote the difference $\mathbf{x}^{out} - \mathbf{x}^{in}$ by $\Delta\mathbf{x}$. Analogously we define $\Delta\mathbf{y}$ and $\Delta\mathbf{z}$. The idea is to find the furthest point from the core point, feasible to the LP-relaxation of $Proj_{x,y,z}\mathcal{F}(MC)$ and lying on the segment line between the *core point* and the *exterior point*. This point is of the form $(\mathbf{x}^{sep}, \mathbf{y}^{sep}, \mathbf{z}^{sep}) = (\mathbf{x}^{out}, \mathbf{y}^{out}, \mathbf{z}^{out}) - \lambda(\Delta\mathbf{x}, \Delta\mathbf{y}, \Delta\mathbf{z})$. The problem of generating cuts like this reads as follows:

$$(SP - CW)^w \quad \min_{\mathbf{f}, \lambda} \lambda \quad (3.14)$$

$$\text{s.t.} \quad \sum_{a \in \delta_w^+(i)} f_a - \sum_{a \in \delta_w^-(i)} f_a = \begin{cases} z^{out} - \lambda \Delta z, & \text{if } i = w^s, \\ 0, & \text{otherwise,} \end{cases} \quad (3.15)$$

$$f_a + f_{\hat{a}} \leq x_e^{out} - \lambda \Delta x_e, \quad e = \{i, j\} \in E : a = (i, j), \hat{a} = (j, i), \quad (3.16)$$

$$\sum_{a \in A} d_a f_a \leq u z^{out} - u \Delta z \lambda, \quad (3.17)$$

$$0 \leq \lambda \leq 1, \quad (3.18)$$

$$f_a \geq 0, \quad a \in A \quad (3.19)$$

In order to get the Benders feasibility cut we solve its associated dual:

$$(DSP - CW)^w \quad \max_{\alpha, \sigma, v} z^{out} \alpha_{w^s} - \sum_{e \in E} x_e^{out} \sigma_e - u z^{out} v \quad (3.20)$$

$$\text{s.t.} \quad \Delta z \alpha_{w^s} - \sum_{e \in E} \Delta x_e \sigma_e - u \Delta z v \leq 1, \quad (3.21)$$

$$\alpha_i - \alpha_j - \sigma_e - d_a v \leq 0, \quad a = (i, j) \in A : e = \{i, j\},$$

$$\sigma_e, v \geq 0, \quad e \in E$$

Given that $(SP-CW)^w$ is always feasible ($\lambda = 1$ is feasible) and that its optimal value is lower bounded by 0, then, both $(SP - CW)^w$ and $(DSP-CW)^w$ have always finite optimal solutions. Whenever the

optimal value of λ is 0, $(\mathbf{x}^{out}, \mathbf{y}^{out}, \mathbf{z}^{out})$ is feasible. Cuts are added if the optimal value of $(\text{DSP-CW})^w$ is strictly greater than 0. The new cut has the same form as in (3.5). Note that this problem can be seen as a dual normalized version of $(\text{SP})^w$ with the dual constraint (3.21).

Core points for both formulations can be obtained by computing the average of the points described in the proof of Proposition 2 for (MC) and the average of the points in list L obtained by applying Algorithm 2.

3.4. Cut-set inequalities

By projecting out variable vector \mathbf{f} , information regarding the link between vectors \mathbf{x} and \mathbf{z} is lost. *Cut-set inequalities* represent the information lost regarding the connectivity for the O/D pair w in the solution given by design variable vector \mathbf{x} . Let (S, S^C) a (w^s, w^t) -partition of N^w for a fixed O/D pair w . That is (S, S^C) satisfies: i. $w^s \in S$; ii. $w^t \in S^C$, with $S^C = N \setminus S$ its complement. A *cut-set inequalities* is defined as

$$z^w \leq \sum_{\substack{\{i,j\} \in E^w: \\ i \in S, j \in S^C}} x_{\{i,j\}}, \quad p \in P, \quad (S, S^C) \text{ a } (w^s, w^t)\text{-partition of } N^w \quad (3.22)$$

This type of constraints have been studied in several articles, for instance Barahona (1996); Koster et al. (2013); Costa et al. (2009). Note that is easy to see that cut-set inequalities belong to the LP-based Benders family. Let (S, S^C) be a (w^s, w^t) -partition in the graph \mathcal{N}^w for $w \in W$. We take the following dual solution:

- $\alpha_i = 1$ if $i \in S$; $\alpha_i = 0$ if $i \in S^C$.
- $\sigma_e = 1$ if $e = \{i, j\} \in E^w$, $i \in S$, $j \in S^C$; $\sigma_e = 0$, otherwise.
- $v = 0$.

Note that this solution is feasible to $(\text{DSP})^w$ and it induces a cut as in (3.22). In order to improve computational performance, we test two approaches to include these inequalities:

1. We implement a modification of the Benders callback algorithm with the following idea. First, for each $w \in W$, using the solution vector (\mathbf{x}, \mathbf{y}) from the master, the algorithm generates a network (N^w, E^w) with capacity 1 for each edge built. Then, a Depth-First Search (DFS) algorithm is applied to obtain the connected component containing w^s . If the connected component does not contain w^t , a cut in the form (3.22) is added. Otherwise, we generate a Benders cut as before. This routine is depicted in Algorithm 3.

We tested this implementation with subproblems in the form $(\text{DSP-CW})^w$. We observe that by using Algorithm 3 with CW the convergence is slower and we generate more cuts. This might

Algorithm 3 Callback implementation with cut-set inequalities.

Require: $(x_e, e \in E, z^w, w \in W)$ from the master vector solution $(\mathbf{x}, \mathbf{y}, \mathbf{z})$.

for $w \in W$ **do**

Build graph $(N^w(\mathbf{x}), E^w(\mathbf{x}))$ induced by the solution vector \mathbf{x} from the master.

Compute the connected component S in $(N^w(\mathbf{x}), E^w(\mathbf{x}))$ containing w^s .

if w^t is included in S **then**

Add the cut $z^w \leq \sum_{\substack{\{i,j\} \in E^w \\ i \in S, j \in S^c}} x_{\{i,j\}}$

else

Solve the corresponding subproblem $((\text{DSP})^w, (\text{DNSP})^w, (\text{DSP-CW})^w)$ and add cut if it is necessary.

end if

end for

return Cut.

be due to the fact that these cuts do not include information about the length of the path in the graph, but only information regarding the existence of the path. These preliminary results are shown in Table 4, which shows average values obtained for solution times in seconds and the number of cuts needed.

Network	BD_CW		Algorithm 3+BD_CW	
	t	cuts	t	cuts
N10	0.23	48	0.15	46
N20	2.47	411	2.53	500
N40	619.31	3486	722.02	3554

Table 4: Comparing the performance of the Algorithm 3. N10, N20 and N40 refer to networks with 10, 20 and 40 nodes respectively.

2. We add to the *Master Problem* the *cut-set inequalities* at the origin and at the destination of each O/D pair $w \in W$ at the beginning of the algorithm. In particular, this valid inequalities has the form:

$$\begin{cases} z^w \leq \sum_{e \in \delta(w^s)} x_e, \\ z^w \leq \sum_{e \in \delta(w^t)} x_e, \end{cases} \quad (3.23)$$

This means that for each O/D pair to be covered, there should exist at least one edge incident

to its origin and one edge incident to its destination, i.e. each O/D pair should have at least one arc going out of its origin and another one coming in its destination.

4. Computational Results

In this section, we compare the performance of the different families of *Benders cuts* presented in Section 3 using the branch-and-Benders-cut algorithm (noted **B&BC**).

4.1. Data sets: benchmark networks and random instances

We divide the tested instances into two groups: *benchmarks instances* and *random instances*. Our *benchmarks instances* are composed by the Sevilla García-Archilla et al. (2013) and Sioux networks ”Hellman (Accessed June 16th, 2020).

The Sevilla instance is composed partially by the real data given by the authors of García-Archilla et al. (2013). From this data, we have used the topology of the underlying network, cost and distance vector for each arc and the demand matrix. This network is composed of 49 nodes and 119 edges. Originally, the set of O/D pairs W was formed by all possible ones ($49 \cdot 48 = 2352$). However, some entries in the demand matrix of this instance are 0 and we exclude them from the analysis. We consider as private utility u twice the shortest path in the underlying network. Each node cost is generated according to an uniform distribution $\mathcal{U}(2000, 4000)$. The available budget has been fixed as 30% of the cost of building the whole underlying network and the minimum demand to be covered as $\beta = 0.5$.

For the Sioux instance, the topology of the network is described by 24 nodes and 38 edges. Set W is also formed by all possible O/D pairs ($38 \cdot 37 = 1406$). With respect to the parameters, they have been chosen in the same manner as for *random instances*.

We generate our *random instances* as follows. We consider planar networks with a set of n nodes, with $n \in \{10, 20, 40, 60\}$. Nodes are placed in a grid of n square cells, each one of 10 units side. For each cell, a point is randomly generated close to the center of the cell. For each setting of nodes we consider a planar graph with its maximum number of edges, deleting each edge with probability 0.3. We replicated this procedure 10 times for each n , so that the number of nodes is the same while that the number of edges may vary. Therefore, there are 40 different underlying networks. We name these instances as $N10$, $N20$, $N40$ and $N60$. We provide the average cycle availability, connectivity and density for *random instances* networks in Table 5. A couple of them are depicted in Figure 2.

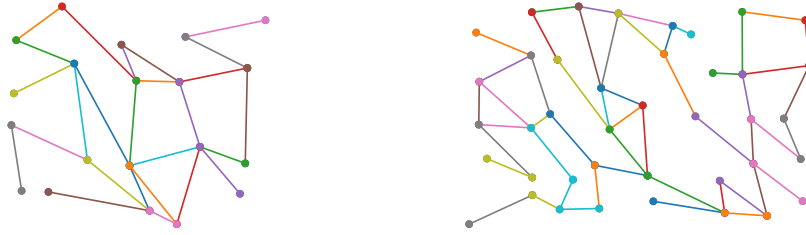


Figure 2: Example of underlying networks with $|N|=20$ and $|N|=40$.

Network	Cycle availability	Connectivity	Density
	$\frac{ E - N +1}{2 N -5}$	$\frac{ E }{ N }$	$\frac{ E }{3(N -2)}$
N10	0.11	1.05	0.44
N20	0.11	1.12	0.41
N40	0.13	1.22	0.43
N60	0.16	1.29	0.45
Overall	0.12	1.17	0.43

Table 5: Cycle availability, connectivity and density parameters for the underlying networks in *random instances*.

Construction costs b_i , $i \in N$, are randomly generated according to a uniform distribution $\mathcal{U}(7, 13)$. Then, each node costs 10 monetary units on average. Construction cost of each edge $e \in E$, c_e , is set to its Euclidean length. These two parameters are rounded to integer numbers. It means that building the links cost 1 monetary unit per length unit. We set C_{max} equal to 50% of the cost of building the whole underlying network considered. We denote this total cost as TC , so $C_{max} = 0.5TC$.

To build set W , we randomly pick each possible O/D pair of nodes with probability 0.5. In consequence, this set has $\frac{n(n-1)}{2}$ pairs on average. Parameter u^w is set to 2 times the length of the shortest path between w^s and w^t . We name this shortest distance SP^w . Finally, the demand g^w for each O/D pair w is randomly generated according to the uniform distribution $\mathcal{U}(10, 300)$.

4.2. Algorithmic setting

Our preliminary experiments have shown that including cuts only in integer nodes is more efficient than including them in nodes with fractional solutions. Thus, in our experiments we only separate integer solutions unless we specify the opposite. We study the different implementations of B&BC proposed in Sections Sections 3.1, 3.2 and 3.3. We name them with the following nomenclature:

- **BD_Trnd**: B&BC algorithm using the feasibility subproblems structure $(DSP)^w$, and its corresponding feasibility cuts (3.5).

- **BD_Norm**: B&BC algorithm using the normalized subproblems structure $(\text{DNSP})^w$, and its corresponding cuts (3.10) and (3.11).
- **BD_CW**: B&BC algorithm using the subproblems structure $(\text{DSP} - \text{CW})^w$, and feasibility cuts (3.5).

We compare our algorithms with the direct use of **CPLEX**, and the automatic benders proposed by **CPLEX**, noted by **AUTO_BD**. **CPLEX** provides different implementations of its automatic Benders depending on the information that the user provides to the solver: i. **CPLEX** attempts to decompose the model strictly according to the decomposition provided by the user; ii. **CPLEX** decomposes the model by using this information as a hint and then refines the decomposition whenever possible; iii. **CPLEX** automatically decomposes the model, ignoring any information supplied by the user. We tested these three possible settings, and the only competitive was the first one.

Furthermore we tested the following features:

- **CS**: If we include *cut-set inequalities* at each origin and destination as in (3.23).
- **IS**: If we provide an initial solution to the solver.
- **RNC**: If we add Benders cuts at the root node.

4.3. Performance of algorithms on random instances

All the experiments were performed with one hour CPU time limit. Tables in this section show average values obtained for solution times in seconds, percent relative gaps, and number of cuts needed. We consider in the average only the instances solved at optimality for all the algorithms.

First, we compare the performance of **CPLEX** for formulations (MC) and (PC) and the different B&BC implementations. All the algorithms are able to solve at optimality instances N10 and N20 in less than 7 seconds for (MC) and (PC). Instances in set N40 were not all solved at optimality neither for (MC) nor (PC) (see first block rows in Table 7 and Table 11). For (MC) the fastest algorithm was **BD_CW** in sets N10, N20 and N40 for the instances solved at optimality. This does not happen for (PC), since we can observe that **AUTO_BD** is slightly faster. **AUTO_BD** solved more instances for both, (MC) and (PC). This trend is confirmed in (MC) for instances in set N60 where the optimality gap obtained after one hour is smaller in **AUTO_BD** as it is shown in Table 8. However, for (PC) the gap after one hour for **BD_CW** is slightly better than the other methods in this family (see Table 12).

Now we refer to the effect in the performance by including **CS**. In general, solution times decrease as the amount of cuts required as well when **CS** is considered. The only exception occurs with **BD_Norm** for (MC) where the computing time is slightly larger. Despite the fact that **BD_CW** generates a larger amount of cuts, this is the most efficient method in terms of computational times for (MC) and (PC) in families N10, N20 and N40. In contrast to the case without **CS**, by adding these initial cuts **BD_Trde**,

BD_Norm and BD_CW are able to solve all the instances in N40 within the time limit for both problems. For problems in set N60, we also have better gaps after one hour in comparison with AUTO_BD. This difference is significant in (MC) with a reduction of more than 5%, but it is less significant for (PC).

For N60 we compare the performance by setting an initial feasible solution IS and adding cuts at the root node RNC. We perform this experiment by computing the optimality gap after one hour limit. First we note that we obtain worse solutions by adding RNC in both problems with all the algorithms tested. For (MC) we observe that adding an initial solution is only profitable for BD_CW+CS, obtaining in average a 3.5% better optimality gap. The impact of adding an initial solution for (PC) is significant for BD_Trld+CS, BD_Norm+CS and BD_CW+CS obtaining in average solutions with a gap around 4% smaller. This improvement is not significant for BD_Auto. In summary, in the set of instances N60 we have that the best algorithms are BD_CW+CS+IS for (MC) and BD_CW+CS+IS and BD_Norm+CS+IS for (PC).

Network	CPLEX	Auto_BD		BD_Trld		BD_Norm		BD_CW	
	t	t	cuts	t	cuts	t	cuts	t	cuts
N10	0.18	0.43	27	0.25	92	0.24	91	0.19	94
N20	6.77	4.51	273	3.89	620	3.18	590	3.34	641
N40	1646.93	617.85	1967	1095.25	3990	541.03	3677	457.81	4137

Network	Auto_BD+CS		BD_Trld+CS		BD_Norm+CS		BD_CW +CS	
	t	cuts	t	cuts	t	cuts	t	cuts
N10	0.32	12	0.21	49	0.28	52	0.23	54
N20	3.94	178	2.29	382	2.50	383	1.85	416
N40	484.95	1248	637.49	2378	575.87	2530	272.39	3186

Table 6: Comparing the performance of the three algorithms for (MC).

	CPLEX	Auto_BD	BD_Trld	BD_Norm	BD_CW
without CS	3	10	9	8	8
+CS	-	10	10	10	10

Table 7: Instances N40 solved for (MC) within a time limit of 1 hour.

In the following, we analyze the performance of algorithms BD_Norm+CS BD_CW+CS when changing parameters C_{max} , β and u in the corresponding models. In Tables 14 and 15, we report average solution times and cuts needed to obtain optimal solutions for N40 for different values of these parameters. The instances are grouped by the three different increasing values of the available budget C_{max} (Table 14.a) or β (Table 15.a) and private utility u (Tables 14.b and 15.b). It is observed that for (MC) the

	Auto_BD		BD_TrD		BD_Norm		BD_CW	
	gap	cuts	gap	cuts	gap	cuts	gap	cuts
without CS	38.54	6545	45.68	14068	44.53	13340	43.77	16707
+CS	30.06	3729	24.27	8754	22.17	8912	25.76	11378

Table 8: Computing gaps to solve N60 (MC) instances comparing the performance of three families of Benders cuts.

	AUTO_BD+CS+IS		BD_TrD+CS+IS		BD_Norm+CS+IS		BD_CW+CS+IS	
	gap	cuts	gap	cuts	gap	cuts	gap	cuts
without RNC	32.90	4987	27.23	9038	26.94	9469	22.27	11151
+RNC	-		37.88	8054	37.92	8230	33.58	10834

Table 9: Computing gaps to solve N60 (MC) instances comparing the performance of three families of Benders cuts.

Network	CPLEX	Auto_BD		BD_TrD		BD_Norm		BD_CW	
	t	t	cuts	t	cuts	t	cuts	t	cuts
N10	0.1765	0.29	16	0.24	92	0.28	89	0.20	91
N20	6.7281	4.87	305	3.55	607	4.68	681	2.15	606
N40	2153.1458	504.06	1752	657.59	4470	514.42	4246	837.41	4412

Network	Auto_BD+CS		BD_TrD+CS		BD_Norm+CS		BD_CW+CS	
	t	cuts	t	cuts	t	cuts	t	cuts
N10	0.28	11	0.16	56	0.20	57	0.145	54
N20	4.12	213	3.11	497	3.43	495	2.070	461
N40	439.23	1527	261.74	3528	323.21	3583	197.55	3949

Table 10: Comparing the performance of the three algorithms for (PC).

	CPLEX	Auto_BD	BD_TrD	BD_Norm	BD_CW
without CS	3	9	8	8	8
+CS	-	10	10	10	10

Table 11: Instances N40 solved for (MC) within a time limit of 1 hour.

	Auto_BD		BD_TrD		BD_Norm		BD_CW	
	gap	cuts	gap	cuts	gap	cuts	gap	cuts
without CS	20.49	7009	20.40	14784	21.41	15501	19.93	15116
+CS	15.92	5109	14.89	12354	14.09	11687	14.50	11744

Table 12: Computing gaps to solve N60 (PC) instances comparing the performance of three families of Benders cuts

	AUTO_BD+CS+IS		BD_Trld+CS+IS		BD_Norm+CS+IS		BD_CW+CS+IS	
	gap	cuts	gap	cuts	gap	cuts	gap	cuts
without RNC	15.86	4372	11.06	8961	10.47	8490	10.44	9683
+RNC	-	-	20.93	10971	21.28	11449	19.94	11053

Table 13: Computing gaps to solve N60 (PC) instances comparing the performance of three families of Benders cuts.

bigger the values of C_{max} the shorter is the average solution time. Table 14.b. shows that the larger is the parameter u the shorter is the solution time for BD_Norm+CS. This behavior seems to be different if we are using BD_CW+CS, which takes less time if the difference between both types of transport is smaller or larger than $2SP$.

C_{max}	BD_Norm+CS		BD_CW+CS		u	BD_Norm+CS		BD_CW+CS	
	t	cuts	t	cuts		t	cuts	t	cuts
0.3TC	1053.56	1580	873.58	2017	1.5SP	802.05	2792	495.84	3041
0.5TC	622.45	2634	375.30	3358	2SP	622.46	2634	375.30	3358
0.7TC	151.24	3970	177.90	5035	3SP	591.02	2674	490.28	3173

a. b.

Table 14: Sensitivity analysis for (MC) with $|N| = 40$.

For (PC), Table 15.a shows that for $\beta = 0.7$ both algorithms take less time to solve at optimality in comparison with $\beta = 0.3$ and $\beta = 0.5$. BD_CW+CS is 5 minutes faster in average than BD_Norm+CS with $\beta = 0.5$. For $\beta = 0.3$ the result is the opposite, BD_Norm+CS is 100 seconds faster in average than BD_CW+CS. By varying u , we observe that the less the difference between public and private mode distances in the underlying network, the longer it will take to get optimality.

β	BD_Norm+CS		BD_CW+CS		u	BD_Norm+CS		BD_CW+CS	
	t	cuts	t	cuts		t	cuts	t	cuts
0.3	640.28	2675	744.95	2848	1.5SP	653.47	3625	620.79	3613
0.5	697.87	3673	387.40	3914	2SP	697.87	3673	387.40	3914
0.7	273.53	3873	242.04	4460	3SP	561.43	3521	378.11	3643

a. b.

Table 15: Sensitivity analysis for (PC) with $|N| = 40$.

4.4. Performance of algorithms in benchmark instances

We start analyzing the Sevilla instance. Tables 17 and 18 show some results for this instance solved with BD_CW+CS. Based on this case, figures in Tables 17 and 18 show the solution graphs for

different parameter values. Points not connected in these graphs refer to those nodes that have not been built. The O/D pairs involving some of these nodes are thus not covered. They have been drawn to represent these not covered areas. Data corresponding to each case are collected at the bottom of its figure, in which $v(\text{ILP})$ refer to objective value. For model (MC), parameter `cost` represents the cost of the network built, and, for (PC), Z makes reference to the demand covered. For (MC) we see that the solution times vary from 21 seconds to 2243 seconds depending on the parameters C_{max} and u . For (PC) these times are in the range of 353 seconds to 1358 seconds. We observe that smaller values of C_{max} carry bigger solution times as in *random instances*. We also observe that, as opposite to *random instances*, higher values of β are translated in larger solution times. Besides, in this instance, for both models, the shorter is the parameter u the larger are the solution times.

Furthermore, we compare the performance of the GRASP algorithm from García-Archilla et al. (2013) and our implementation BD_CW+CS. We implemented the GRASP algorithm to run 5 times and return the best solution. Table 16 shows solution times, best value for GRASP (**Best Value**), the optimality gap, and the optimal value computed with BD_CW+CS. On the one hand, we observed that the more time BD_CW+CS takes to compute the optimal solution the larger is the gap of the solution returned by GRASP. This happens for smaller values of the budget C_{max} and utility u . On the other hand, for problems where GRASP obtains small optimality gap, BD_CW+CS is more efficient to compute the optimal solution. In other words, since GRASP is a constructive algorithm, it is not competitive for instances whose optimal solution captures most of the demand.

Finally we discuss the results for the Sioux instance. They have also been obtained by using BD_CW+CS. We observe for (MC), as in the Sevilla network, that smaller values of C_{max} and u the larger solution times. The same is true varying β in (PC), but not for u . It takes less time if the difference between both modes of transport is smaller or larger than $2SP$. These results are summarized in Tables B.19 and B.20 in Appendix B.

Our exact method is able to get the best quality solution, with a certificate of optimality in reasonable times. Given that network design problems are strategic decisions, having the best quality decision is often more important than the computational times. However, having efficient exact methods as the proposed in this article, allows decision makers to perform sensitivity analysis with optimality guarantees in reasonable times.

5. Conclusions

In this article, we have studied two variants of the *Network Design Problem: Maximal Covering Network Design Problem* where we maximize the population covered under a budget constraint; and *Partial Set Covering Network Design Problem* where the total building cost is minimized satisfying

C_{max}	u	GRASP			BD_CW+CS	
		t	Best Value	gap	t	v(ILP)
0.2TC		110.829	48629	6.97	1036.11	52274
0.3TC	2SP	260.220	59828	3.96	313.07	62294
0.4TC		396.226	63546	0.72	21.36	64011
0.3TC	1.5SP	267.275	55778	6.97	2243.83	59958
	3SP	225.312	62049	0.99	113.88	62670

Table 16: Sensitivity analysis for GRASP algorithm García-Archilla et al. (2013) with Sevilla instance.

a lower bound in the total population covered. We state integer programming formulations that are stronger than existing ones for both problems. We provide some polyhedral properties of these formulations useful from the algorithmic point of view. We develop exact methods based on Benders decomposition. We also discuss some preprocessing routines to scale-up the instances solved. This preprocessing techniques play a key role in order to get information about the instances and to get better algorithmic performance. Our computational results show that the techniques developed in this article allow to obtain better solutions in less time than the techniques in the existing literature.

Acknowledgments

Víctor Bucarey and Martine Labbé have been partially supported by the Fonds de la Recherche Scientifique - FNRS under Grant(s) no PDR T0098.18. Natividad González-Blanco and Juan A. Mesa are partially supported by Ministerio de Economía y Competitividad (Spain)/FEDER under grant MTM2015-67706-P.

Bibliography

- Balas, E., & Perregaard, M. (2002). Lift-and-project for mixed 0–1 programming: recent progress. *Discrete Applied Mathematics*, *123*, 129–154.
- Balas, E., & Perregaard, M. (2003). A precise correspondence between lift-and-project cuts, simple disjunctive cuts, and mixed integer gomory cuts for 0-1 programming. *Mathematical Programming*, *94*, 221–245.
- Barahona, F. (1996). Network design using cut inequalities. *SIAM Journal on optimization*, *6*, 823–837.
- Botton, Q., Fortz, B., Gouveia, L., & Poss, M. (2013). Benders decomposition for the hop-constrained survivable network design problem. *INFORMS journal on computing*, *25*, 13–26.

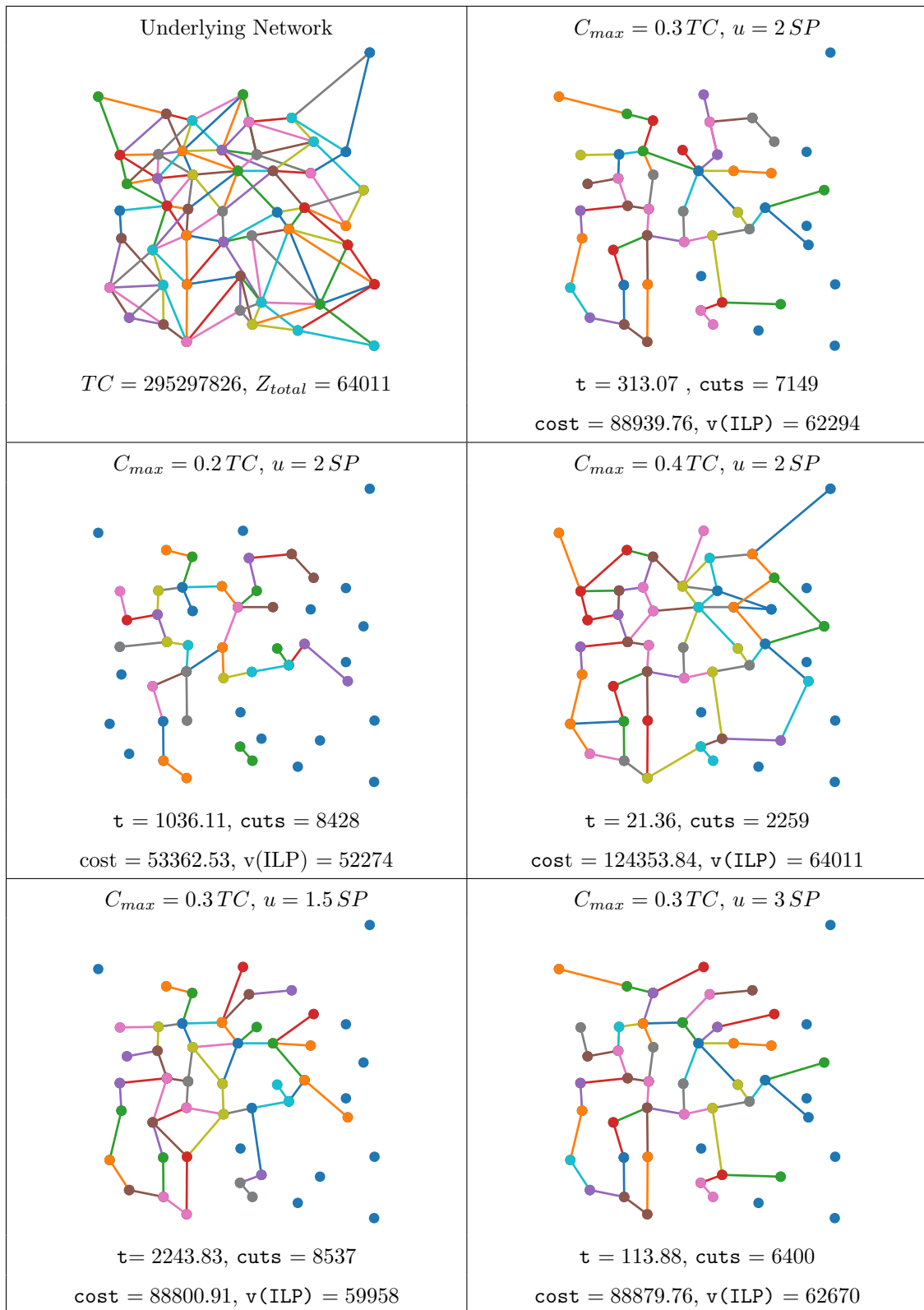


Table 17: Sensitivity analysis Sevilla Network with (MC).

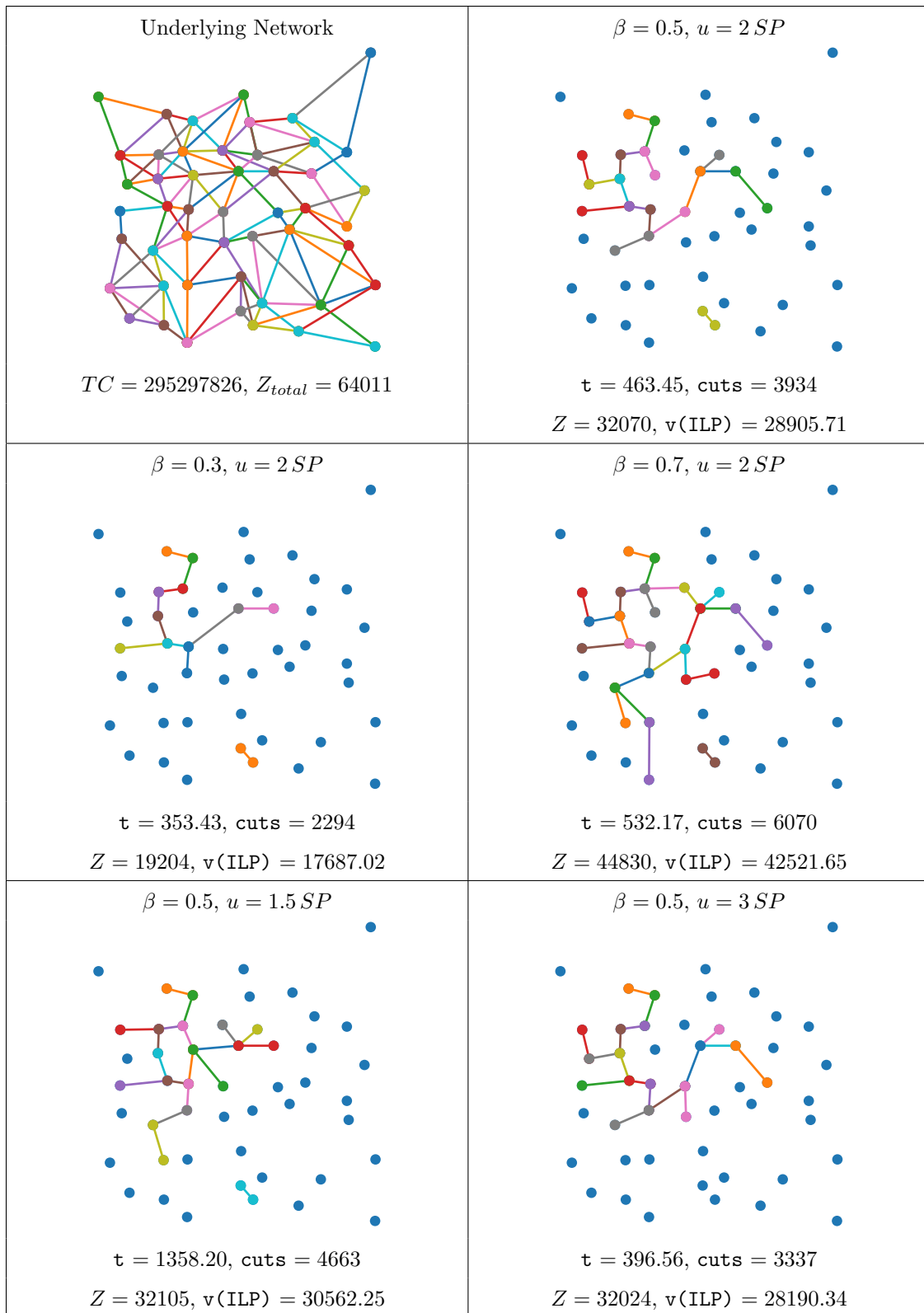


Table 18: Sensitivity analysis Sevilla Network with (PC).

- Canca, D., De-Los-Santos, A., Laporte, G., & Mesa, J. A. (2017). An adaptive neighborhood search metaheuristic for the integrated railway rapid transit network design and line planning problem. *Computers & Operations Research*, *78*, 1–14.
- Canca, D., De-Los-Santos, A., Laporte, G., & Mesa, J. A. (2019). Integrated railway rapid transit network design and line planning problem with maximum profit. *Transportation Research Part E: Logistics and Transportation Review*, *127*, 1–30.
- Cascetta, E. (2009). *Transportation systems analysis: models and applications* volume 29. Springer Science & Business Media.
- Church, R., & ReVelle, C. (1974). The maximal covering location problem. In *Papers of the regional science association* (pp. 101–118). Springer-Verlag volume 32.
- Conforti, M., & Wolsey, L. A. (2019). “Facet” separation with one linear program. *Mathematical Programming*, *178*, 361–380.
- Cordeau, J.-F., Furini, F., & Ljubić, I. (2019). Benders decomposition for very large scale partial set covering and maximal covering location problems. *European Journal of Operational Research*, *275*, 882–896.
- Costa, A. M., Cordeau, J.-F., & Gendron, B. (2009). Benders, metric and cutset inequalities for multicommodity capacitated network design. *Computational Optimization and Applications*, *42*, 371–392.
- Desrochers, M. (1988). *An algorithm for the shortest path problem with resource constraints*. École des hautes études commerciales, Groupe d’études et de recherche en
- Fischetti, M., Salvagnin, D., & Zanette, A. (2010). A note on the selection of benders’ cuts. *Mathematical Programming*, *124*, 175–182.
- Fortz, B., & Poss, M. (2009). An improved benders decomposition applied to a multi-layer network design problem. *Operations Research Letters*, *37*, 359 – 364. URL: <http://www.sciencedirect.com/science/article/pii/S016763770900073X>. doi:<https://doi.org/10.1016/j.orl.2009.05.007>.
- García-Archilla, B., Lozano, A. J., Mesa, J. A., & Perea, F. (2013). Grasp algorithms for the robust railway network design problem. *Journal of Heuristics*, *19*, 399–422.
- Guihaire, V., & Hao, J.-K. (2008). Transit network design and scheduling: A global review. *Transportation Research Part A: Policy and Practice*, *42*, 1251–1273.

- "Hellman, F. (Accessed June 16th, 2020). *Sioux Falls Variants for Network Design*. URL: http://www.bgu.ac.il/~bargera/tntp/SiouxFalls_CNDP/SiouxFallsVariantsForNetworkDesign.html.
- Koster, A., Phan, T. K., & Tieves, M. (2013). Extended cutset inequalities for the network power consumption problem. *Electronic Notes in Discrete Mathematics*, *41*, 69–76.
- Król, A., & Król, M. (2019). The design of a metro network using a genetic algorithm. *Applied Sciences*, *9*, 433.
- Ljubić, I., Putz, P., & Salazar-González, J.-J. (2012). Exact approaches to the single-source network loading problem. *Networks*, *59*, 89–106.
- Magnanti, T. L., Mireault, P., & Wong, R. T. (1986). Tailoring benders decomposition for uncapacitated network design. In *Netflow at Pisa* (pp. 112–154). Springer.
- Magnanti, T. L., & Wong, R. T. (1981). Accelerating benders decomposition: Algorithmic enhancement and model selection criteria. *Operations research*, *29*, 464–484.
- Marín, Á. G., & Jaramillo, P. (2009). Urban rapid transit network design: accelerated benders decomposition. *Annals of Operations Research*, *169*, 35–53.
- Perea, F., Menezes, M. B., Mesa, J. A., & Rubio-Del-Rey, F. (2020). Transportation infrastructure network design in the presence of modal competition: computational complexity classification and a genetic algorithm. *TOP*, (pp. 1–33).
- Rahmaniani, R., Crainic, T. G., Gendreau, M., & Rei, W. (2017). The benders decomposition algorithm: A literature review. *European Journal of Operational Research*, *259*, 801–817.
- Schmidt, M., & Schöbel, A. (2014). Location of speed-up subnetworks. *Annals of Operations Research*, *223*, 379–401.

Appendix A. Pseudo-code for initial feasible solutions

In this section we provide the pseudo-codes to get an initial feasible solution for (MC) and (PC) described in Section 2.4. We denote as N_s, E_s and W_s the set of indices of design and mode choice variables set to 1 at the end of each algorithm.

Algorithm 4 Initial Feasible Solution for (MC)

Initialization: Set $N_s = \emptyset, E_s = \emptyset$ and $W_s = \emptyset$ and $IC = 0$.

Compute ratio $r_w = \frac{g^w}{C(\text{Path}_w)}$:

for $w \in W$ in decreasing order of r_w **do**

$$\bar{C} = C(\text{Path}_w) - \sum_{e \in E_s \cap \tilde{E}^w} c_e - \sum_{i \in N_s \cap \tilde{N}^w} b_i.$$

if $IC + \bar{C} \leq C_{max}$ **then**

$$W_s \leftarrow W_s \cup \{w\}.$$

$$E_s \leftarrow E_s \cup \tilde{E}^w.$$

$$N_s \leftarrow N_s \cup \tilde{N}^w.$$

$$IC \leftarrow IC + \bar{C}.$$

end if

end for

$x_e = 1$ for $e \in E_s$, 0 otherwise.

$y_i = 1$ for $i \in N_s$, 0 otherwise.

$z^w = 1$ for $w \in W_s$, 0 otherwise.

return (x, y, z)

Algorithm 5 Initial Feasible Solution for (PC)

Initialization: Set $\bar{W}_s = W$ and $Z_s = Z_{total}$.

Compute ratio $r_w = \frac{g^w}{C(\text{Path}_w)}$:

for $w \in W$ in decreasing order of r_w **do**

if $Z_s - g^w \geq \beta Z_{total}$ **then**

$$W_s \leftarrow W_s \setminus \{w\}.$$

$$Z_s \leftarrow Z_s - g^w.$$

end if

end for

$x_e = 1$ if $e \in \bigcup_{w \in W_s} \tilde{E}^w$, 0 otherwise.

$y_i = 1$ if $i \in \bigcup_{w \in W_s} \tilde{N}^w$, 0 otherwise.

$z^w = 1$ for $w \in W_s$, 0 otherwise.

return (x, y, z)

Appendix B. Results for SIOUX networks

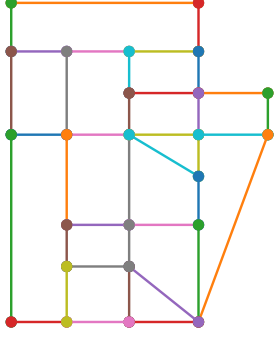
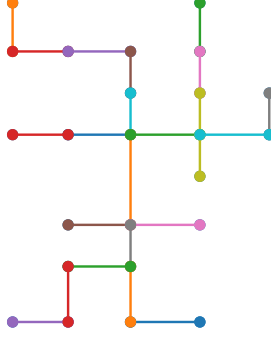
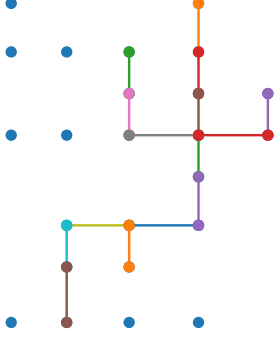
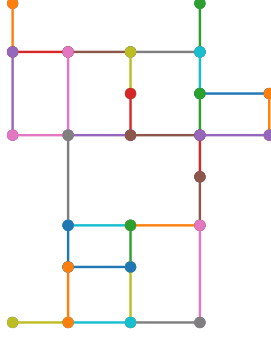
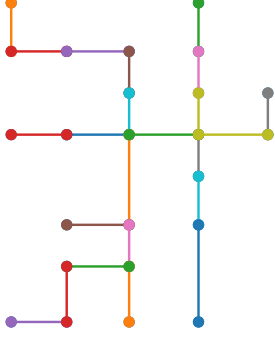
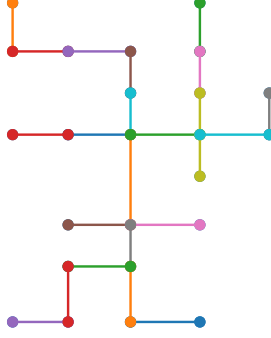
<p>Underlying Network</p>  <p>$TC = 4171, Z_{total} = 84437$</p>	<p>$C_{max} = 0.5TC, u = 2SP$</p>  <p>$t = 22.85, \text{cuts} = 3496$ $\text{cost} = 2070, v(\text{ILP}) = 75488$</p>
<p>$C_{max} = 0.3TC, u = 2SP$</p>  <p>$t = 458.84, \text{cuts} = 3056$ $\text{cost} = 1237, v(\text{ILP}) = 35039$</p>	<p>$C_{max} = 0.7TC, u = 2SP$</p>  <p>$t = 2.73, \text{cuts} = 801$ $\text{cost} = 2870, v(\text{ILP}) = 82699$</p>
<p>$C_{max} = 0.5TC, u = 1.5SP$</p>  <p>$t = 60.31, \text{cuts} = 3460$ $\text{cost} = 2080, v(\text{ILP}) = 68227$</p>	<p>$C_{max} = 0.5TC, u = 3SP$</p>  <p>$t = 14.17, \text{cuts} = 2641$ $\text{cost} = 2070, v(\text{ILP}) = 75488$</p>

Table B.19: Sensitivity analysis Sioux Network with (MC).

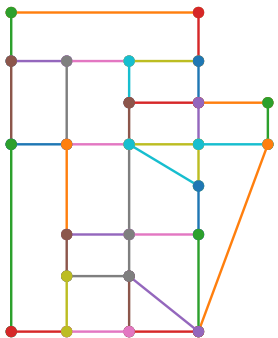
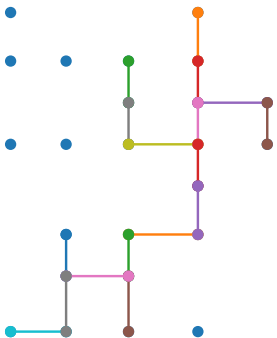
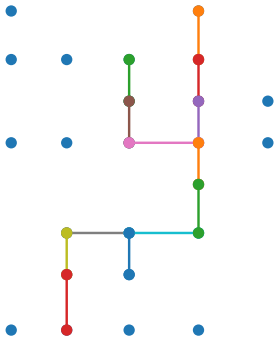
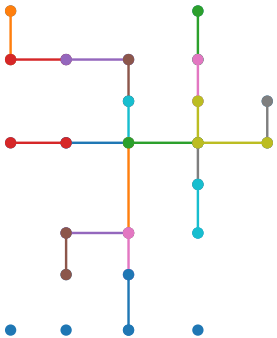
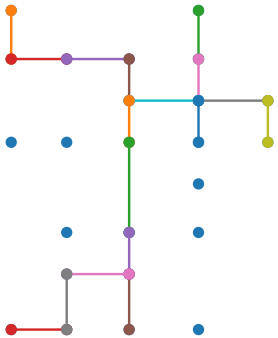
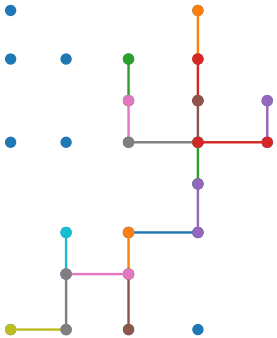
<p>Underlying Network</p>  <p>$TC = 4171, Z_{total} = 84437$</p>	<p>$\beta = 0.5, u = 2 SP$</p>  <p>$t = 429.85, \text{cuts} = 3306$ $Z = 44112, v(\text{ILP}) = 1411$</p>
<p>$\beta = 0.3, u = 2 SP$</p>  <p>$t = 925.68, \text{cuts} = 2783$ $Z = 24588, v(\text{ILP}) = 1058$</p>	<p>$\beta = 0.7, u = 2 SP$</p>  <p>$t = 136.06, \text{cuts} = 3674$ $Z = 60276, v(\text{ILP}) = 1726$</p>
<p>$\beta = 0.5, u = 1.5 SP$</p>  <p>$t = 1471.84, \text{cuts} = 3793$ $Z = 43599, v(\text{ILP}) = 1491$</p>	<p>$\beta = 0.5, u = 3 SP$</p>  <p>$t = 1149.26, \text{cuts} = 3128$ $Z = 42331, v(\text{ILP}) = 1411$</p>

Table B.20: Sensitivity analysis Sioux Network with (PC).