Computers and Operations Research 81 (2017) 178–191

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor

Designing and constructing networks under uncertainty in the construction stage: Definition and exact algorithmic approach

Eduardo Álvarez-Miranda[a,*], Jordi Pereira[b]

[a] Department of Industrial Engineering, Universidad de Talca, Curicó, Chile
[b] Department of Engineering and Sciences, Universidad Adolfo Ibáñez, Viña del Mar, Chile

ARTICLE INFO

Article history:
Received 16 June 2015
Revised 31 May 2016
Accepted 22 December 2016
Available online 27 December 2016

Keywords:
Network design
Network construction
Two-stage robust optimization
Exact algorithms

ABSTRACT

The present work proposes a novel Network Optimization problem whose core is to combine both network design and network construction scheduling under uncertainty into a single two-stage robust optimization model. The first-stage decisions correspond to those of a classical network design problem, while the second-stage decisions correspond to those of a network construction scheduling problem (NCS) under uncertainty. The resulting problem, which we will refer to as the Two-Stage Robust Network Design and Construction Problem (2SRNDC), aims at providing a modeling framework in which the design decision not only depends on the design costs (e.g., distances) but also on the corresponding construction plan (e.g., time to provide service to costumers). We provide motivations, mixed integer programming formulations, and an exact algorithm for the 2SRNDC. Experimental results on a large set of instances show the effectiveness of the model in providing robust solutions, and the capability of the proposed algorithm to provide good solutions in reasonable running times.

1. Introduction and motivation

The construction of transportation network systems usually consists of two stages: (i) the strategic network design stage, in which the structure and the composition of the network are defined; and (ii) the construction planning stage, in which a schedule of the construction process in a given time span is established. In other words, in the first stage, the edges of the network to be constructed are *optimally* selected (along with attributes such as capacities), and in the second stage, a schedule for constructing the edges in accordance with the available resources is chosen.

The problems appearing in the first stage fall within the classification of typical Network Design (ND) problems, and a broad body of literature is available (for a recent reference see [22]). Classical examples of ND problems are the minimum spanning tree, MST, problem and the Steiner Tree problem. ND decisions are usually made taking into account data such as geographical distances. Such data is known with complete certainty at the moment of the decision taking, and they are unlikely to change in later stages.

Conversely, the Network Construction Scheduling (NCS) problem of the second stage has been scarcely investigated (see [3]). However, the NCS deserves attention since it defines how and when different elements of the network become available (i.e., how and when the users can access the network) and has a significant impact on the resource requirements of the construction plan. Consequently, NCS affects service revenues as well as construction costs and, therefore, the intrinsic uncertainty of the data (weather conditions, labor efficiency, supplies availability) needs to be taken into account.

Moreover, in order to provide a *reasonable* performance under different situations, including adverse conditions, we believe that network construction must consider the network design and the construction schedule decisions jointly. This approach allows to avoid significant inefficiencies during construction caused by the strict minimization of the network design costs.

Note that the previous setting is not the only situation in which coordinating network design and construction planning provides significant improvements over their separate resolution. For instance, networks subject to frequent disruptions (e.g. disasters) are likely to benefit from considering the tactical decisions (reconstruction of the network after disruptions) during the design process. In such a case, the network should be designed weighting construction costs, possible reconstruction costs, and the cost associated to the deprivation of service until reconstruction. Therefore, the inherent uncertainty associated to disasters must be accounted for.

Two well-known approaches for dealing with uncertainty in optimization are Two-Stage Stochastic Optimization (2SSO) and

* Corresponding author.
E-mail address: ealvarez@utalca.cl (E. Álvarez-Miranda).

http://dx.doi.org/10.1016/j.cor.2016.12.018
0305-0548/

Robust Optimization (RO). In 2SSO (see [9]) the solutions are built in two stages. In the first stage, a *partial* collection of decisions is defined and completed later on (in the second stage), when the actual data is revealed. Hence, the objective is to minimize the cost of the initial (first-stage) decisions plus the *expected* cost of the recourse (second-stage) decisions. The quality of the solutions provided by this model strongly depends on the accuracy of the random representation of the parameter values (such as probability distributions) that estimate the second-stage expected cost. Nonetheless, sometimes such accuracy is not available, so the use of RO models dealing with *deterministic uncertainty* arises as a suitable alternative (see [7,8,17]). On the one hand, these models do not require assumptions about the distribution of the uncertain input parameters; but on the other hand, they are usually meant for calculating single-stage decisions that are immune (in a sense) to all possible realizations of the parameter values.

A novel alternative that combines RO and 2SSO is Two-stage Robust Optimization (2SRO). As in RO, no stochasticity of the parameters is assumed and, as in 2SSO, decisions are taken in two stages. In this case, the cost of the second-stage decision is computed by looking at the worst-case realization of the data. Therefore, the goal of 2SRO is to find a *robust* first-stage solution that minimizes both the first-stage cost plus the worst-case second-stage cost among all possible data outcomes. 2SRO constitutes a rather generic classification of models; for references on different 2SRO settings we refer the reader to [6,23].

The core of this paper is to combine both network design and network construction under uncertainty into a 2SRO model. The first-stage decisions correspond to those of a classical ND problem, while the second-stage decisions correspond to those of a NCS under uncertainty. The resulting problem, which we will refer to as the Two-Stage Robust Network Design and Construction Problem (2SRNDC), aims at providing a modeling framework in which the design decision not only depends on the design costs (e.g. distances), but also on the corresponding construction plan. In other words, an optimal first-stage designed network, say N^*, not only ensures efficiency with respect to the design cost but it also ensures that the induced second-stage construction cost will be, in the worst case, as cheap as possible.

Among the possible 2SRNDC formulations, this paper focuses on the case in which a spanning tree has to be constructed and the scheduling decisions try to minimize the time before users have access to the network. The scheduling objective is represented as the sum of connection times to a central node (e.g. depot or service provider). The resulting problem will be referred to as the Two-stage Robust Spanning Tree Sum of Connection Times problem (2SRSTSCT).

1.1. Our contribution

The contribution of this paper is threefold. First, we contribute to the new field of problems dealing with network construction scheduling by providing a model that combines both stages of decision (design and construction). Second, we analyze the uncertainty introduced by construction, departing from the classical approach that associates uncertainty to network design. This is a novel approach as the source of uncertainty introduces a secondary optimization problem instead of splitting the decision in two decision stages. And third, we provide an algorithmic framework to tackle this new type of problems. Such a framework has been tailored for a particular version of the problem, but this exposition stresses that the proposed algorithm can be adapted to other variants with a different first-stage network design problem or a different second-stage scheduling problem. The version of the problem considered in this work corresponds to essential versions of both the first and the second stage. The first stage selects a spanning tree. The second stage orders the construction of the edges of the spanning tree while minimizing the average connection time from a central node (the depot or service provider) to the nodes of the network (customers). The formulation of the scheduling problem has been shown to resemble a single machine scheduling problem with sum of completion times objective (see [3]).

1.2. Previous work

We now provide a brief review of recent publications dealing with ND combined with multi-period scheduling decisions and with ND combined with 2SRO.

Network construction scheduling problems. The model studied in [21] corresponds to an infrastructure restoring problem in which the goal is to locate a set of work groups to install additional arcs into a disrupted network, with the aim of enabling the flow of a commodity from supply to demand nodes. A Mixed Integer Programming (MIP) formulation is proposed, reduction techniques are devised, heuristic procedures are designed, and extensive numerical results on a set of realistic instances are reported.

In [5,13,15] the authors study the problem of *completing* a network (mainly by adding new edges) in T periods ensuring that in each period $t \in \{1, \ldots, T\}$ an optimally designed sub-network, $N^*(t)$, is obtained from the network completed up to that period. In [13], a minimum spanning tree must be solved at each period; in [5], a shortest path has to be found at every t. Finally, in [15], $N^*(t)$ corresponds to an allocation of flow that ensures a maximum flow between a fixed pair of nodes. Formulations, heuristic and approximation algorithms, reduction techniques, and computational results are reported.

A more recent general framework for integrating network design and scheduling decisions is proposed in [20]. This new approach generalizes the model previously proposed in [21]. In this case, at each period, the performance of the network expansion strategy is evaluated with respect to different network models: maximum flow, minimum-cost flow, shortest path, and spanning tree. Moreover, two types of objective functions are considered: a *cumulative* objective (which aims at optimizing the weighted network performance over the time span), and a *threshold* objective (which aims at minimizing the time needed to *reach* the predefined performance value).

Another alternative to combine network design and scheduling decisions corresponds to the Network Construction Scheduling Problem (NCS) defined in [3]. In the NCS, the edges of a network need to be constructed in order to provide connectivity between a node (the depot), and the remaining nodes (the customers). The objective is to obtain a construction order for the selected edges that optimizes a metric associated to the time required to connect a given node (the service provider) to every other node of the network.

This formulation allows the use of different scheduling objectives, like the minimization of the weighted or unweighted sum of completion times (equivalent to minimizing average connection times) [3], the minimization of the maximum lateness (equivalent to minimizing the maximum delay with respect to the due date of each node), or the minimization of the number of tardy nodes (equivalent to minimizing the total number of unfulfilled contractual dates) [4]. These objectives describe problems in which the connection time between a source of service (the depot) and the remaining nodes (the customers) is critical.

In this work the scheduling decision falls within the NCS framework, and we consider the minimization of the average time required to connect the nodes, which is equivalent to the unweighted sum of completion times. This objective constitutes a valid construction performance when no additional information is available

or when the connection of each node is considered equally important.

The practical pertinence of addressing the construction phase of transportation and supply network systems has been stressed in all the aforementioned papers. However, previous work has not considered the uncertainty associated to construction times in the construction phase. Furthermore, scheduling decisions after disruptions or over long periods of time are likely to be affected by uncertainties that should be incorporated in order to obtain an appropriate scheduling planning. This paper is, to the best of our knowledge, the first attempt to provide a decision framework for the design and construction scheduling of networks with uncertainty in the later phase.

Two-stage robust optimization for network design. Robust Network design models involving decisions taken in two stages are not new. The common setting considers that the design process is split into two phases: a partial network is designed in a first phase and, once the uncertainty is revealed in the second phase, the design is completed. Examples of Two-stage Robust Optimization for Network Design can be found in [11] (where formulations and algorithms for different variants of the two-stage shortest path problem are given), and [1] (where the authors present properties and exact algorithms for robust two-level network design problems). Further examples can be found in [2]. In these problems, uncertainty, which is typically modeled by means of scenarios or interval data, is associated with the design costs or the composition of the network.

The decision making setting discussed in this work differs from the setting of the aforementioned papers. Here, the second stage corresponds to ascertaining the construction order of the network designed in the first stage, and the uncertainty is associated with the construction times.

2. The two-sage robust network design and construction problem

2.1. General framework: the 2SRNDC

Let us assume an existing or potential network $G = (V, E)$, which is considered to be *incomplete* and, therefore, it does not operate according to some given requirements. One can say that the network is initially incomplete or it became incomplete due to a natural disaster, a massive failure produced by an intentional or unintentional phenomenon, demographic changes, etc. New connections (edges) can be *constructed* within (or *added* to) the network, and/or the capacity of some of the existing connections can be *increased*. Both the construction of new connections and the capacity expansion of the existing ones have to first be designed according to some performance criterion $f(\cdot)$ and topological and/or operative requirements represented by $\mathcal{X}(G)$. This results into an *expansion policy* embedded in a network $\mathbf{X}$. Once this network has been designed, decision makers need to establish *how* $\mathbf{X}$ will be constructed along the time span T. Since the expansion decisions are to be implemented in a later stage, it is natural to think that the cost of this implementation is not currently known with complete certainty. An alternative to modeling uncertainty in such a context is using a set of discrete scenarios Ω, in which each scenario $\omega \in \Omega$ is characterized by a given realization of expansion resources (time, cost, etc.). For each scenario $\omega \in \Omega$, the construction scheduling decisions $\mathbf{Y}^\omega$ must be decided considering a performance criterion $g(\cdot, \omega, \mathbf{X})$ and satisfying a set of requirements represented by $\mathcal{Y}(G, \mathbf{X})$.

Our goal is to find a network design policy $\mathbf{X}^*$ such that

$$\mathbf{X}^* = \arg \min_{\mathbf{X} \in \mathcal{X}(G^0)} \left\{ f(\mathbf{X}) + \max_{\omega \in \Omega} \min_{\mathbf{Y}^\omega \in \mathcal{Y}(G^0, \mathbf{X})} g(\mathbf{Y}^\omega, \omega, \mathbf{X}) \right\}. \quad \text{(RNDC)}$$

The policy defined by $\mathbf{X}^*$ not only ensures efficiency with respect to the design cost (due to the minimization of the first-stage objective), but it also ensures that the induced second-stage construction scheduling cost will be, in the worst case, as cheap as possible.

The robust network design construction problem (RNDC) is quite general, since the expression $\mathcal{X}(G)$ may restrict the solution to be a spanning tree, a Steiner tree, a Hamiltonian tour or any other network topology. Likewise, the constraints embodied by $\mathcal{Y}(G, \mathbf{X})$ can impose many different conditions to the construction schedule, and similar statements can be made for the functions $f(\mathbf{X})$ and $g(\mathbf{Y}^\omega, \omega, \mathbf{X})$. The particular variant of the RNDC considered throughout this paper is described and formulated in the remainder of this section.

2.2. The 2SRSTSCT: definition

The problem addressed in this paper forces $\mathbf{X}$ to be a spanning tree in G, and forces $\mathbf{Y}^\omega$, $\forall \omega \in \Omega$, to be an order for constructing the $n-1$ connections encompassing $\mathbf{X}$. The performance of $\mathbf{X}$ is calculated as the sum of the design cost of each of the edges that comprise it, while the performance of $\mathbf{Y}^\omega$ is calculated as the total *completion* time, i.e., the sum of the time that each node in V needed to *wait* before being connected to a specific node, the depot. This version of the (RNDC) will be coined Two-stage Robust Spanning Tree Sum of Connection Times problem (2SRSTSCT). We now provide an MIP formulation for this version of the 2SRSTSCT.

We are given a set of nodes V that must be connected to a given root node $r \in V$ (the depot), using a subset of the edges defined by the set of potential connections, E. There is a design cost function, $d : E \to \mathbb{R}^+$, such that d_e corresponds to the cost of using edge $e \in E$ to establish the connected network.

Let $\mathbf{x} \in \{0, 1\}^{|E|}$ be a binary vector such that $x_e = 1$ if edge $e \in E$ belongs to a spanning tree of G, and $x_e = 0$ otherwise. Therefore, the set $\mathcal{X}(G)$ can be defined as

$$\mathcal{X}(G) = \{\mathbf{x} \in \{0, 1\}^{|e|} \mid \mathbf{x} \text{ induces a spanning tree on } G\}; \quad (X)$$

hence, a spanning tree T on G is defined as $T \equiv \{e \in E \mid x_e = 1\}$. For a given $\mathbf{x} \in \mathcal{X}(G)$, the cost design function is

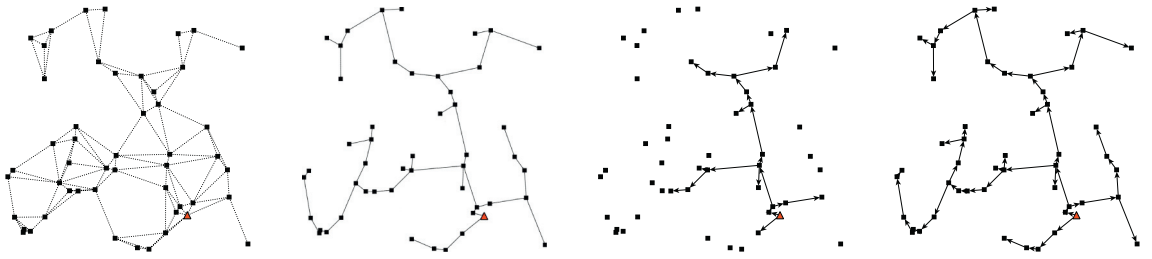
$$f(\mathbf{x}) = \sum_{e \in E} d_e x_e.$$

For a given first-stage feasible spanning tree encoded by $\mathbf{x} \in \mathcal{X}(G)$, and a given scenario $\omega \in \Omega$, the second-stage problem corresponds to the NCS. In other words, it corresponds to a construction scheduling for $\mathbf{x}$ with minimum average completion time; such problem can be formulated as follows.

Let $\mathbf{y} = \left(\mathbf{y}^1, \ldots, \mathbf{y}^\omega, \ldots, \mathbf{y}^{|\Omega|}\right)$ be a collection of binary vectors $\mathbf{y}^\omega \in \{0, 1\}^{|E| \times (|V|-1)}$, defined as follows. For each $\omega \in \Omega$ and $k = \{1, \ldots, |V| - 1\}$, $y_{ij}^{k\omega} = 1$ if edge $e \in E$ is the kth constructed arc when scenario $\omega \in \Omega$ is realized, and $y_{ij}^{k\omega} = 0$ otherwise. There is a time construction function $c : E \times \Omega \to \mathbb{R}^+$ such that c_e^ω corresponds to the time for constructing edge $e \in E$ if scenario $\omega \in \Omega$ is realized.

For a given scenario $\omega \in \Omega$ and a given designed network $\mathbf{x}' \in \mathcal{X}(G)$, a feasible construction scheduling, induced by a vector $\mathbf{y}^\omega$, must be established such that the edges of $\mathbf{x}'$ are constructed sequentially and the first constructed edge is connected to the root r. Therefore, the set $\mathcal{Y}(G, \omega, \mathbf{x}')$ can be defined as:

$$\mathcal{Y}(G, \omega, \mathbf{x}') = \{\mathbf{y}^\omega \in \{0, 1\}^{|E| \times (|V|-1)} \mid \mathbf{y}^\omega \text{ is an ordered completion of } \mathbf{x}' \text{ starting from } r\} \quad (y)$$



(a) Example of G (b) Example of $\mathbf{x}'$ (c) First 20 constructed links (d) Complete construction of $\mathbf{x}'$

Fig. 1. Examples of an instance and solutions of the 2SRSTSCT.

for every $\omega \in \Omega$. For a given $\mathbf{y}^{\omega} \in \mathcal{Y}(G, \omega, \mathbf{x}')$, the construction cost under scenario ω is provided by

$$g(\mathbf{y}^{\omega}, \omega, \mathbf{x}') = \frac{1}{|V|-1} \sum_{k=1}^{|V|-1} \sum_{e \in E|_{x'_e=1}} (|V|-k) c_e^{\omega} y_e^{k\omega}.$$

Notice that $\mathbf{y}^{\omega}$, $\omega \in \Omega$, represents a sequence of edges. After the construction of each edge, an additional node is connected to the root node r and, thus, the construction time of the kth constructed edge contributes to the connection time of $(|V|-k)$ nodes. Consequently, the above function computes the average connection time of the nodes in V.

For a better understanding of the problem, let us examine Fig. 1. Fig. 1(a) shows an example of G, i.e., an incomplete graph in which no connection has been established yet (only *potential* connections are known and they are displayed by dotted lines). Fig. 1(b) shows an example of the designed network. As required by the model, $\mathbf{x}'$ induces a spanning tree on G, in which the node represented by a triangle corresponds to the root node. For a fixed scenario, say $\omega' \in \Omega$, Fig. 1(c) shows the network constructed up to the 20th connection, i.e., with 20 built edges, allowing to connect 20 nodes to the root (there are still 29 nodes *waiting* to be reached). Finally, Fig. 1(d) shows the completely built network, which coincides with $\mathbf{x}'$.

An MIP formulation for the 2SRSTSCT is established by the following set of constraints that combine (X) and (y) and impose the minimization of the worst-case performance among different scenarios,

$$Z_{\text{RSTC}} = \min\ f(\mathbf{x}) + \theta \tag{RSTC.1}$$

$$\text{s.t.}\quad \theta \geq g(\mathbf{y}^{\omega}, \omega, \mathbf{x}), \forall \omega \in \Omega \tag{RSTC.2}$$

$$\mathbf{x} \in \mathcal{X}(G). \tag{RSTC.4}$$

$$\mathbf{y}^{\omega} \in \mathcal{Y}(G, \omega, \mathbf{x}), \forall \omega \in \Omega \tag{RSTC.3}$$

Objective (RSTC.1) minimizes the sum of network design cost plus the cost of the second-stage for the worst case scenario, θ, calculated in (RSTC.2). (RSTC.4) defines the domain of the first-stage variables $\mathbf{x}$ and (RSTC.3) the domain of the second-stage variables $\mathbf{y}$.

It is possible to define an MIP model of (RSTC.1)–(RSTC.4), with a full compact description of (RSTC.2) and (RSTC.3). This formulation will be referred to as *extended formulation*, and it is provided in Appendix A.1. The proposed model is based on a directed representation of G, therefore, the first-stage variables $\mathbf{x}$ are associated to arcs instead of edges. This allows the use of the so-called *directed cut-set* inequalities to describe (RSTC.4), although such constraints are exponential in number, they enable the design of efficient separation procedures (see Section 3).

Please note that while the previous formulation corresponds to a specific selection of first- and second-stage problems, the framework described by the (RNDC) is sufficiently general to consider alternative topologies, $\mathcal{X}$ and $\mathcal{Y}$, and objectives in each of the stages, $f(\cdot)$ and $g(\cdot)$.

2.3. The 2SRSTSCT: relations with other problems

The 2SRSTSCT is equivalent to several other combinatorial optimization problems under some conditions. These relationships are used in the development of the proposed solution method and also provide complexity results for the problem in hand.

First, if the second-stage costs are negligible or the construction costs of the edges are identical, then Problem 2SRSTSCT is equivalent to the problem of finding a minimum spanning tree.

Second, if there is one scenario for the construction stage, the first-stage costs are negligible, then Problem 2SRSTSCT is equivalent to the flowtime network construction problem (FNCP) discussed in [3]. The FNCP has been shown to be strongly NP-hard using a reduction from the SET COVER problem (see [3]) and, thus, the 2SRSTSCT is also strongly NP-hard.

Third, if the network is a tree and there is one scenario for the construction stage, then the 2SRSTSCT is equivalent to minimizing the total completion of jobs (the edges) with out-tree precedence constraints (the path to the depot) in a single machine. The aforementioned problem can be solved in $O(n\log n)$, (see [10], pp. 73–77).

The first and the third relationships are used in the heuristic generation of additional inequalities, Section 3.2, and in the primal heuristic, Section 3.3.

The second relationship identifies the NP-hardness status of the problem and it is used in the matheuristic generation of inequalities described in Section 3.2.

3. Algorithmic framework

The extended formulation described in the previous section allows to solve the problem within a branch-and-cut algorithm. Such an algorithm is based on the separation of the cut-set inequalities (RSTC.4) underlying $\mathbf{x} \in \mathcal{X}(G)$ (see , e.g., [16], for further details). Preliminary computational results showed that this approach is ineffective even for small size instances. This is mainly due to the poor quality of the linear relaxation of the second-stage problem embodied by (RSTC.2) and (RSTC.3).

Therefore, in order to solve the 2SRSTSCT in a more effective manner, we put forward an alternative approach that avoids the use of second-stage variables and constraints by iteratively introducing constraints that describe the corresponding second-stage function, θ. The resulting branch-and-cut algorithm relies only on first-stage variables and combines both types of inequalities.

In the remainder of this section, we outline the novel features of the proposed method.

3.1. Approximation of the second-stage objective

For a given solution $\mathbf{x}' \in \mathcal{X}(G)$ at a given node of the branch-and-cut search tree, let $\omega' \in \Omega$ be the scenario associated with the worst-case second-stage objective $\theta(\mathbf{x}')$ (*binding* scenario), and let $\mathbf{y}'^{\omega'}$ be the corresponding optimal schedule. Using this notation, and according to the formulation (RSTC.1)–(RSTC.4), the constraint

$$\theta \geq \sum_{k=1}^{|V|-1} \sum_{e \in E} (|V| - k) c_e^{\omega'} y'^{k\omega'}_e x_e, \qquad (\theta\text{-C})$$

ensures a valid lower bound on θ. This constraint guarantees that if $\mathbf{x}'$ corresponds to the optimal solution, then θ should be $\theta(\mathbf{x}')$ (see [18], for further details).

For any $\mathbf{x}'$, constructing (θ-C) requires the coefficients $\mathbf{y}'^{\omega'}$, which correspond to the optimal solution of the NCS problem for the binding scenario. One can efficiently solve the NCS based on the following observation:

Observation 1. (See [3]) For a fixed solution $\mathbf{x}' \in \mathcal{X}(G)$ and a given scenario $\omega \in \Omega$, the corresponding NCS problem, $\min\{g(\mathbf{y}^{\omega}, \omega, \mathbf{x}') \mid \mathbf{y}^{\omega} \in \mathcal{Y}(G, \omega, \mathbf{x}')\}$, can be solved in $\mathcal{O}(n \log n)$ time.

Proof. Consider that the edges $e \in E\,|_{x'_e=1}$ are jobs and the values c_e^{ω} are the processing times. The tree induced by $\mathbf{x}'$ defines natural out-tree precedence constraints for the jobs. Minimizing the total completion time of n jobs with out-tree precedence constraints can be done in $\mathcal{O}(n \log n)$ time (see [10] pp. 73–77). □

Clearly, for any given $\mathbf{x}'$, (θ-C) can be ascertained in $\mathcal{O}(|\Omega| n \log n)$ time using Observation 1 by solving the corresponding NCS for each of the $|\Omega|$ scenarios and selecting the binding one. This corresponds to the basic scheme for generating (θ-C) inequalities, and it works only if the solution, $\mathbf{x}'$, induces a spanning tree.

Note that, at the beginning of the optimization process, the model is initialized with the constraint induced by the solution of the minimum spanning tree on G.

3.2. Heuristic generation of (θ-C) constraints

As already said, the previous approach is valid only when the solution, $\mathbf{x}'$, defines a feasible first-stage solution. In order to improve the overall performance of the algorithm, we propose the early identification of first-stage solutions whose corresponding (θ-C) inequalities are likely to define tight bounds on θ. Such schemes are described below.

Rounding heuristic. At a given node of the branch-and-cut search tree, let $\tilde{\mathbf{x}}$ be the corresponding solution of the linear relaxation problem (LP). If $\tilde{\mathbf{x}}$ is fractional, one can still generate a valid (θ-C) constraint by heuristically rounding $\tilde{\mathbf{x}}$. To do so, we first solve the MST on G with edge costs $\tilde{\mathbf{c}}$ defined by

$$\tilde{c}_e = c_e(1 - \tilde{x}_e), \ \forall e \in E,$$

and, afterwards, the corresponding (θ-C) constraint is added to the model.

These additional constraints are likely to be very similar to previously generated ones. Therefore, we impose a condition such that the constraint induced by $\tilde{\mathbf{x}}'$ is added if $\tilde{\mathbf{x}}'$ is *orthogonal* with respect to the current incumbent solution, say $\check{\mathbf{x}}$. Concretely, we add the inequality induced by $\tilde{\mathbf{x}}'$ if the following equation holds:

$$\Delta(\tilde{\mathbf{x}}', \check{\mathbf{x}}) = \sum_{e \in E} \left| \tilde{x}'_e - \check{x}_e \right| \geq \nu,$$

i.e., $\tilde{x}'_e$ and $\check{x}_e$ must induce spanning trees that differ in at least ν edges. This ensures that the new inequality is more likely to cut off a solution that lies in a *different* area of the search space. In the long run, this accelerates the overall performance of the algorithm, since the search space is *shrunk* more effectively.

Matheuristic (θ-C)-generation. Similar to the previous case, we have designed a technique for generating cuts when $\tilde{\mathbf{x}}$ is *partially* fractional. These inequalities are based on: (i) optimally finding the worst-case NCS on a partial graph induced by $\tilde{\mathbf{x}}$, say $\tilde{G}(\tilde{\mathbf{x}})$; (ii) approximating the worst-case NCS on the remaining graph $G \setminus \tilde{G}(\tilde{\mathbf{x}})$; and (iii) constructing a valid inequality for the original problem. Hence, one can think about the proposed constraint generation method as a mathheuristic constraint generator in which a partial solution is extended to create a complete solution by optimizing its second-stage cost.

The matheuristic-based method can be described as follows:

1. First, identify $T(\tilde{\mathbf{x}})$, the set of edges of the connected component rooted at the depot verifying $\tilde{x}_e = 1$.
2. Second, for each scenario $\omega \in \Omega$, solve the NCS of the partial tree induced by $T(\tilde{\mathbf{x}})$ (using the result in Observation 1).
3. Third, *collapse* $T(\tilde{\mathbf{x}})$ and map it into a single node $r_{T(\tilde{\mathbf{x}})}$. For each scenario ω, the optimal NCS solution for the remaining graph (with $r_{T(\tilde{\mathbf{x}})}$ as root node) is computed using a branch-and-bound procedure as in [3].
4. Fourth, for each scenario, let $\mathbf{y}^{\omega}$ be the solution obtained by merging the construction order found in Step 2 with the construction order found in Step 3 for scenario ω.
5. Finally, find the binding scenario ω' induced by the solutions obtained in Step 4, and check whether the corresponding (θ-C) constraint is satisfied before its inclusion in the current model.

The previous method is able to generate good heuristic solutions under the assumption that those arcs already fixed to integrality by the LP are likely to be very important (and thus, they need to be scheduled earlier). Step 3 requires the resolution of an NP-hard problem for each scenario. Hence, the applicability of the method is highly related to the ability to optimally solve the remaining subproblem, and, therefore, we only solve the problem when the number of nodes *not* spanned by $T(\tilde{\mathbf{x}})$ is equal to or smaller than some predefined constant μ.

In Step 3, the sub-problem is solved using a branch-and-bound method that explores the search tree using a depth-first search exploration scheme. A solution is represented as a sequence identifying the order in which nodes are reached. A branching decision corresponds to appending a node to the sequence. Note that the edge used to reach a node corresponds to the shortest edge that connects it to an already reached node. In order to prune the search tree, the lower bounds and dominance rules proposed in [3] are used. Furthermore, the implementation uses some additional dominance rules, which are described below. Please note that the current branch-and-bound does not make use of the mixed exploration scheme proposed in [3] in order to reduce the memory required by the matheuristic.

The new dominance rules consider the relation between the scheduling part of the problem and the one machine total completion time scheduling problem, which can be optimally solved using the Shortest Processing Time (SPT) rule. As noted in [3], the solution of the second-stage optimization problem corresponds to a sequence of nodes in connection order. The edge used to connect a node corresponds to the smallest-cost edge connecting a previously connected node and the newly connected one; hence, the selected edges induce a spanning tree. Notice that the second-stage optimization problem can be seen as a one machine total completion time scheduling problem in which the processing time of the

jobs (all of the nodes with the exception of the depot) depends on the preceding jobs in the sequence. Under some conditions, the SPT rule still applies to each partial ordering of jobs. Based on the previous observation, it is possible to derive two properties that one optimal sequence S $(v_1 = 1, v_2, \ldots, v_{|V|-1})$ is to fulfill.

Proposition 1. *There is an optimal sequence S in which each pair of consecutive nodes, v_i and v_{i+1}, fulfils at least one of the two following conditions: (i) the cost of the edge used to connect node v_i is smaller than the cost of the edge used to connect node v_{i+1}; or (ii) the other endpoint of the edge used to connect node v_{i+1} is node v_i.*

Proof. The first condition corresponds to the SPT rule, and can be proved using exchange arguments, see [12]. Condition (ii) introduces an exception in order to account for the additional characteristics of the problem. □

Proposition 2. *There is an optimal sequence S in which the cost of the edge used to connect node v_i is lower than the cost of the edge used to connect node v_j (with $i < j$), unless the other endpoint of the edge used to connect node v_j is one of the nodes $v_{i+1}, \ldots, v_{j-1}$ or node v_i is the endpoint of an edge used to connect some node in $v_{i+1}, \ldots, v_{j-1}$.*

Proof. If the exception does not hold, an exchange between nodes v_i and v_j reduces the total cost without changing the cost of the edges used to connect any of the nodes. Hence, the exchange argument suffices to proof the proposition. □

These propositions are verified during the branch-and-bound tree by reducing the list of candidate nodes to append to the sequence to those nodes that fulfill the above conditions. It is also possible to verify that some nodes that do not belong to the sequence under construction cannot fulfill Proposition 1. If such a node is identified, the partial solution under construction is not considered for branching.

The application of these propositions, in addition to the previous dominance rules and lower bounds described in [3], allows us to solve instances with up to 40 nodes in less than a second.

3.3. Further enhancements

Branching rule. In order to further improve the lower bounds, we designed an ad-hoc branching strategy. We assume again that $\tilde{\mathbf{x}}$ is the LP solution at the current node of the branch-and-cut tree; let $G(\tilde{\mathbf{x}})$ be the connected subgraph, rooted at the depot, induced by the edges e verifying $\tilde{x}_e = 1$. Let $\delta(G(\tilde{\mathbf{x}}))$ be the set of edges with an endpoint in $G(\tilde{\mathbf{x}})$ and the other endpoint outside the subgraph. Our branching rule forces to branch on $x_{(kl)^*}$ variable, where edge e: $\{k, l\}^*$ corresponds to

$$\{k, l\}^* = \arg \min_{\{k,l\} \in \delta(G(\tilde{\mathbf{x}}))} |\tilde{x}_{k,l} - 0.5|.$$

Intuitively, this branching rule helps to define search branches characterized by disjoint connected components rooted at the depot, which are likely to yield a more efficient enumeration and, therefore, a much better performance of the algorithm.

Primal heuristic. So far, the proposed enhancements mainly focus on improving the lower bounds along the optimization process. The final ingredient of our algorithm is a primal heuristic that helps to find new incumbent (primal) solutions and push down the upper bounds quickly. Although several ideas were tried, the simplest one turned out to be the most effective.

The implemented idea follows: (i) Given a fractional solution $\tilde{\mathbf{x}}$, find a *near* integer vector $\tilde{\mathbf{x}}'$ by solving an MST on G using $\tilde{\mathbf{c}}$; (ii) calculate $Q(\tilde{\mathbf{x}}')$; and (iii) attempt to set $\left(\tilde{\mathbf{x}}', Q(\tilde{\mathbf{x}}')\right)$ as primal solution. Although extremely simple, this idea was effective in providing reasonable upper bounds in early stages of the optimization process.

4. Computational results

All the experiments were performed on an Intel Core™ i7 (4702QM) 2.2 GHz machine (8 cores) with 16GB RAM. The branch-and-cut was implemented using CPLEX™ 12.5 and the Concert Technology framework. When testing our branch-and-cut, all CPLEX parameters were set to their default values.

Regarding the setting of our algorithm, the constant L (see (θ-C)) was set to 0, and the value of μ was set to 35 nodes. A time limit of 2400 s was imposed in all runs. Two different sets of experiments were conducted. The first set considers randomly generated instances whose topology is similar to street networks. The second set considers instances derived from the Chilean road network.

4.1. Benchmark instances

Randomly generated instances. These are planar random instances, in the Euclidean plane, which were generated following the ideas presented in [14] for the generation of prize collecting Steiner tree instances. The topology of these instances is similar to street networks. The underlying graph is defined as follows: (i) n nodes are randomly located in a one-unit Euclidean square; (ii) an edge e between two nodes i and j is established if the Euclidean distance between them is smaller than $\alpha/\sqrt{n}$, for a fixed $\alpha > 0$, and the graph remains planar.

Once the topology of the network is defined, design costs are set as the Euclidean distances, i.e., $d_e = \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2}$, $\forall e$: $\{i, j\} \in E$, where (x_i, y_i) and (x_j, y_j) are the coordinates of nodes i and j respectively. See Fig. 1 for an example of the network obtained with $n = 50$ and $\alpha = 1.6$.

Each arbitrary scenario of edge construction costs is generated as follows: (i) for each edge $e \in E$, uniformly and randomly generate a number z_e^ω in the interval $\left[0, \sqrt{2}\right]$; (ii) the corresponding construction cost c_e^ω, for each edge $e \in E$, is provided by

$$c_e^\omega = \rho d_e + \sqrt{1 - \rho^2} z_e^\omega,$$

where $\rho \in [0, 1]$ is a user-defined parameter that enables to control the correlation between design and construction costs. This is repeated $|\Omega|$ times, which leads to a set Ω of construction cost scenarios.

Instances of this class are defined by three parameters, n, α and ρ. In particular, we considered $n = \{25, 30, 35, 40, 45, 50, 60, 75, 100\}$, $\alpha = 1.6$ (as suggested in [14]), and $\rho = \{0.00, 0.25, 0.50, 0.75\}$. In our computations, we considered up to 50 scenarios which are created in advance. Therefore, when dealing with instances with 15 scenarios, we simply use the first 15 scenarios out of the 50 scenarios. The same applies, for instance, to 20 scenarios. The scenarios are identical for the different values of all other parameters. By proceeding in this way, it is easier to measure the impact of considering a larger number of scenarios.

Please note that we opted for this scenario generation scheme for the following reasons: (i) it allows us to concentrate on the relationship between the first- and the second-stage costs, given by ρ; (ii) it does not require any additional hypotheses on the way that results can be interpreted; and (iii) it is simple, but general enough, to represent any structure of relationships among scenarios, while generating algorithmically challenging instances.

Instances derived from the Chilean road network. In this case, the underlying graph is given by the interurban road network that connects the cities in Chile (according to the Chilean government, any

Table 1
Comparison between the branch-and-cut approach applied to the extended formulation and the (θ-C)-generation approach.

	Extended formulation			(θ-C)-generation		
Size	# Solved	# Optimal	Av. gap	# Solved	# Optimal	Av. gap
25	23	3	1.96	24	24	0
30	20	2	0.88	24	24	0
35	17	0	2.05	24	23	0.05
40	12	0	4.51	24	8	1.39

locality with more than 5000 inhabitants is considered a city). The resulting network contains 245 nodes and 393 edges. The selected cities are located from the north end, Arica, down to southern city of Punta Arenas, from where most of the connections to the (few) southern most cities are performed via ferries or light aircrafts. This network was selected for the study because Chile endures disasters (specifically earthquakes) regularly, and thus integrating recovery into the network design decision would be beneficial. Furthermore, a subset of the road network was considered in a previous work for the second stage problem [4].

In order to generate smaller instances, we merged nodes using the following procedure: merge into a single node, or *macro city*, any node in which the distance between urban areas is located within P kilometers (and remove edges correspondingly). We considered $P = \{5, 10, 15, 20\}$, which leads to instances with 203 nodes and 297 edges, 152 nodes and 231 edges, 120 nodes and 185 edges, and 91 nodes and 140 edges, respectively. These aggregated instances can be regarded as the design problem, in which the construction of the main communication network between population hubs is solved.

In this case, design costs are taken as the (Euclidean) length of every road segment. Construction cost scenarios are calculated similarly to the case of the ND instances, but the values of z_e^ω are taken from the interval $[0, \max_{e \in E} d_e]$. For these instances, we considered $\rho = \{0.00, 0.25, 0.50, 0.75\}$. The instances were labeled Pkm-CHILE-ρ. Appendix A.3 provides a representation of the network.

4.2. Algorithmic performance

Benefits of the (θ-C)-generation scheme. In order to evaluate the benefits of using the constraint generation scheme with respect to explicitly incorporating the second-stage variables (extended formulation (RSTC.1)–(RSTC.4)), we solved a subset of the randomly generated instances using a branch-and-cut approach that separates *only* the cut-set inequalities (extended formulation approach). Table 1 provides a summary of the comparison between the (θ-C)-generation approach and the extended formulation approach.

For each instance size, we report: (i) the number of instances in which the algorithm was able to provide a solution within the imposed time limit; (ii) the number of optimal solutions found and (iii) the average optimality gaps between the lower and upper bounds provided by the algorithm for the instances in which the algorithm was able to find a solution. The results evidence that the (θ-C)-generation approach outperforms the compact formulation both in number of instances solved and in the average optimality gaps reported. Furthermore, note that the gaps provided by the compact formulation only correspond to the instances in which the algorithm was able to find a solution (the easiest instances). According to the limitations of the compact formulation, the rest of the section only examines the results from the (θ-C)-generation approach.

Influence of algorithmic enhancements. In order to assess the quality of the heuristic and matheuristic cuts, along with the proposed branching rule, we conducted a computational experiment using the set of randomly generated instances. Each instance was solved using eight different versions of the algorithm, one for each combination of presence/absence of a component. The average gap ($\frac{UB-LB}{UB} \cdot 100$) between the reported lower and upper bounds is used as the response variable of the experiment.

The results were analyzed as a repeated measures ANOVA with three factors (the components of the algorithm), see [19], with the main characteristics of the instance (size, correlation, and number of scenarios) as confounding factors. The results of the ANOVA test showed that the introduction of additional matheuristic cuts has a significant impact on the behavior of the algorithm (p-*value* $< 10^{-16}$), while the remaining characteristics are not statistically significant. An analysis of the residuals showed heteroscedasticity issues. Hence, a non-parametric ANOVA test was conducted in order to verify if heteroscedasticity was affecting the conclusions. The test consists in applying the same repeated measures ANOVA on a rank transformation of the response variable. This test provided the same conclusions as the original test. Hence, we are led to conclude that the analysis and its conclusions are satisfactory.

Note that, although the impact of the heuristic cuts and the branching rule is not statistically significant, they do not have a negative effect. Furthermore, we observed that these components provide better solutions for specific instances of the test bed and, thus, we decided to maintain them in the implementation of the algorithm reported in the remainder of the section.

Analysis of the results for random instances. Tables 2 and 3 as well as Fig. 2 provide a summary of the results for the random instances. Table 2 provides an analysis of the results grouped by instance size and number of scenarios (4 instances per group), while Table 3 provides the same results grouped by instance size and correlation between the first-stage and the second-stage costs (6 instances per group). Both tables report the average gap, the average running time and the number of optimal verified solutions (if any).

Additional metrics on the behavior of the algorithm like the average number of cuts generated of each type (cut-sets, (θ-C) constraints, and matheuristically generated (θ-C) constraints) and the average number of explored branch-and-cut tree nodes are provided in Appendix A.2. We do not report the number of CPLEX cuts generated because its number is always very small (the largest number of cuts provided by CPLEX in any tested instance was 5).

Table 2 shows an increase in the average gap as the number of nodes and scenarios grows. Note that, for instances with $n \geq 50$, the optimality of any solution is verified and the average gap is significant, which highlights the difficulty of the problem and shows the limits of the proposed approach to provide and verify optimality of the solutions.

When instances are grouped according to the correlation factor between the first-stage and the second-stage costs, the results show that the average gap grows when the size of the instance and the correlation grows, see Table 3. This pattern is counter-intuitive and is further discussed below.

In order to highlight the influence of the different characteristics of the instances on the solution quality, Fig. 2 reports several boxplots of the evolution of the optimality gap variation according to the characteristics of the instances. Each boxplot represents the average optimality gaps when instances are grouped in accordance with the number of nodes of the network, the correlation between the costs of the first stage and the second stage, or the number of scenarios.

The results show that the algorithm is capable of solving to optimality instances with up to $n = 35$ within the allotted time, and provides near optimal solutions (gaps below 0.05) when $n = 40$, regardless of the remaining characteristics of the instance. For

Table 2
Average gap, average running time of the algorithm for the random instances and number of optimal solutions found (if any). The number of optimal solutions found is reported in parenthesis. Instances are grouped according to size and number of scenarios.

Scen.	5		10		15		20		25		50	
n	Av.gap (# opt)	Av. time	Av.gap (# opt)	Av. time	Av.gap (# opt)	Av. time	Av.gap (# opt)	Av. time	Av.gap (# opt)	Av. time	Av.gap (# opt)	Av. time
25	0.0 (4)	4	0.0 (4)	14	0.0 (4)	21	0.0 (4)	53	0.0 (4)	50	0.0 (4)	105
30	0.0 (4)	12	0.0 (4)	18	0.0 (4)	27	0.0 (4)	58	0.0 (4)	87	0.0 (4)	245
35	0.0 (4)	86	0.0 (4)	282	0.0 (4)	248	0.0 (4)	293	0.0 (4)	542	0.3 (3)	1125
40	0.0 (4)	1597	1.0 (2)	1976	1.0 (2)	2180	1.7	2400	1.8	2400	2.9	2400
50	6.0	2400	7.0	2400	7.9	2400	7.8	2400	8.4	2400	8.9	2400
60	10.7	2400	11.9	2400	11.6	2400	12.9	2400	13.3	2400	14.1	2400
75	17.9	2400	19.1	2400	19.4	2400	19.6	2400	19.8	2400	21.2	2400
100	27.3	2400	28.9	2400	29.3	2400	29.3	2400	30.0	2400	31.6	2400

Table 3
Average gap, average running time of the algorithm for the random instances and number of optimal solutions found (if any). The number of optimal solutions found is reported in parenthesis. Instances are grouped according to size and correlation.

Correlation	0		25		50		75	
n	Av.gap (# opt)	Av. time	Av.gap (# opt)	Av. time	Av.gap (# opt)	Av. time	Av.gap (# opt)	Av. time
25	0.0 (6)	34.3	0.0 (6)	71.5	0.0 (6)	24.2	0.0 (6)	37.2
30	0.0 (6)	66.8	0.0 (6)	92.5	0.0 (6)	99.7	0.0 (6)	41.2
35	0.0 (6)	149.2	0.0 (6)	319.7	0.2 (5)	863.1	0.0 (6)	387.9
40	0.7 (3)	1863.7	0.8 (3)	2079.3	1.8 (1)	2329.8	2.2 (1)	2363.3
45	4.2	2400.0	5.1	2400.0	6.8	2400.0	6.9	2400.0
50	7.2	2400.0	7.5	2400.0	7.9	2400.0	8.1	2400.0
60	11.2	2400.0	12.0	2400.0	13.3	2400.0	13.1	2400.0
75	18.0	2400.0	19.6	2400.0	20.0	2400.0	20.3	2400.0
100	28.0	2400.0	29.6	2400.0	30.5	2400.0	29.4	2400.0

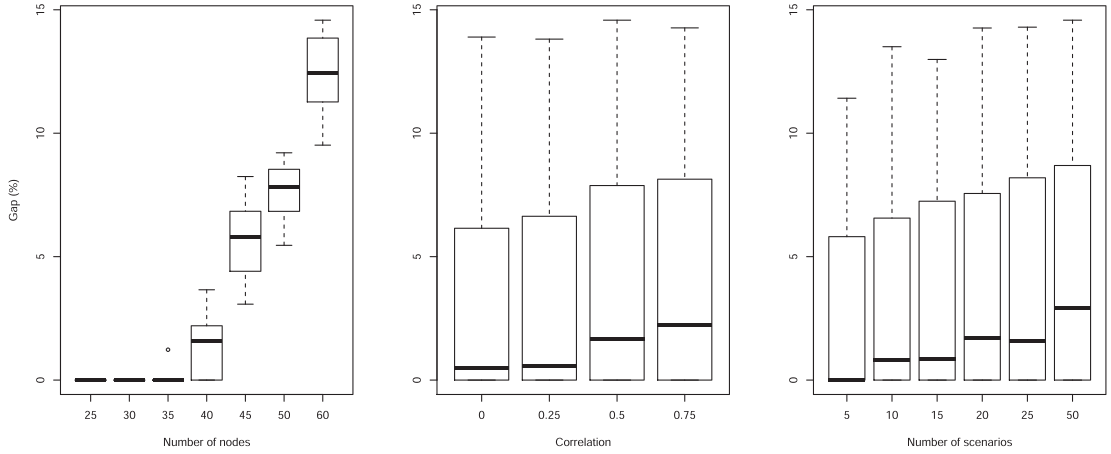


Fig. 2. Optimality gaps for the instances grouped according to their main characteristics (number of nodes, correlation between first-stage and second-stage costs, and number of scenarios.

larger instance sizes, the performance of the algorithm deteriorates.

The remaining characteristics of the instances have a smaller impact on the optimality gaps reported by the algorithm. Note that the correlation between the costs of both the first and second stage, as previously highlighted in Table 3, shows a positive trend (the larger the correlation, the larger the optimality gap). We hypothesize that the perceived behavior can be explained by the way that the underlying network is generated. As one can observe in Fig. 1, there are several 3-cliques in which two edges have approximately the same length, which leads to a similar first-stage design cost. Hence, if the correlation between both stages is low, it is likely that one of the edges will be favored to the detriment of the other, i.e., it is preferred to be constructed in the second stage. Conversely, if the correlation is high, it is more difficult to distinguish between these two edges (they are more *symmetric*), so the algorithm must enumerate more solutions. Please note that this behavior is not observed in the realistic instances discussed in Section 4.4. This is mainly due to the fact that such 3-cliques do not appear frequently.

Also note that the number of scenarios has some impact on the optimality gaps. This is caused by the additional running time required to generate the additional (θ-C) constraints. This phenomenon is further studied when we address the analysis of the solutions in terms of their robustness.

4.3. Analysis of the solutions

The effort for robustness. Increasing the robustness to uncertainty by increasing the number of scenarios entails a trade-off between

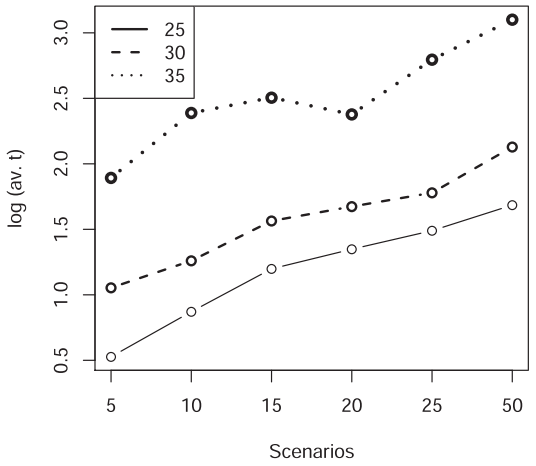


Fig. 3. Average running time.

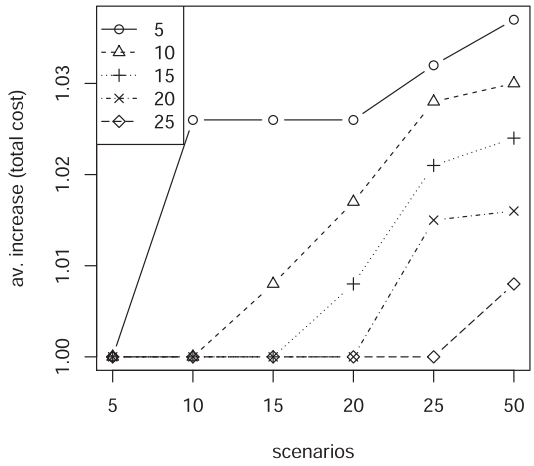


Fig. 4. Average deterioration (increase of the total cost) when the optimal solution for a given number of scenarios is evaluated under an additional set of scenarios ($n = 35$).

the additional robustness provided by these scenarios and the increase in the computational complexity of the problem that needs to be addressed. Therefore, a compromise solution needs to be reached between two opposing objectives: (i) minimizing the computing time required to offer a solution for the problem; and (ii) increasing the protection against uncertainty offered by the solution.

The additional computing requirements associated to increasing robustness have been coined *the effort for robustness* [see 1]. In order to evaluate this effort, Table 2 reports the evolution of the average computing time requirements for instances with different number of scenarios. A visual representation of the computing time metric is provided in Fig. 3.

For different instance sizes, the lines of the graph, and number of scenarios, the X-axis, the average gap required to reach optimality are reported, Y-axis, in logarithmic scale. Results for $n = \{25, 30, 35\}$, which correspond to the instance sizes in which optimality is consistently reached, are reported. Note that, while the increase depicted in the figure roughly follows a line, the scale is logarithmic and, thus, the increase in effort (measured as running time) is exponential to the number of scenarios.

In order to justify the increase in computational requirements, the additional effort required is to be accompanied by greater levels of protection against uncertainty. A possible metric to evaluate this additional protection is to measure how a solution for any given number of scenarios deteriorates as the number of scenarios increases.

Let $Z_{RNDC}^{\Omega_1,\Omega_2}$ be the cost of the optimal solution found under the set of scenarios Ω_1 when evaluated in the set of scenarios Ω_2 ($\Omega_1 \subseteq \Omega_2$). Then, the deterioration ratio $Z_{RNDC}^{\omega_1,\omega_2} / Z_{RNDC}^{\Omega_1,\Omega_1}$ is always equal to or greater than 1, and its value increases as the quality of the solutions found for scenarios Ω_1 deteriorates for some of the scenarios in Ω_2.

Fig. 4 reports the said metric. For all of the instances with $n = 35$, the problem was solved to optimality for each number of scenarios and correlation costs, and the average measure of the deterioration ratio (axis-Y) for the solution obtained using 5, 10, 15, 20, 25 scenarios (each of the different lines) was evaluated on the instances with additional scenarios (axis-X).

Note that the deterioration ratio is always low (below 4% even when the solution for 5 scenarios is evaluated with 50 scenarios). Also note that the trend of deterioration of the solution when the number of scenarios grows, shows that the positive slope is smaller as the number of additional scenarios grows. For example, the deterioration of the metric when increasing from 25 to 50 scenarios is much smaller than when increasing from 20 to 25 scenarios. Consequently, it seems reasonable not to consider larger number of scenarios as their contribution to the robustness of the solution is minimal.

Comparison with the minimum spanning tree solution. Another important question to ponder is how different the solutions for the first-stage problem (construction cost) are to the solutions for the 2SRNDC (construction and recovery costs together). Note that the optimal solution to the first-stage problem corresponds to a minimum spanning tree, MST. As the solution to the problem studied in the first-stage cost is easily solvable, the extra additional effort required to solve the two-stage problem is to be justified by potential differences between the solutions for both problems. In order to illustrate these differences, Fig. 5 represents the ratio between the edges shared by the MST and the best solution found with respect to the number of edges of any spanning tree.

The boxplots show that the difference between the MST and the best solution found by the algorithm increases as the number of nodes increases, and remains relatively constant regardless of the remaining characteristics of the instance. For small-sized problems, this ratio shows a high level of intersection (between 70% and 80% of the edges of both solutions are identical) but the difference increases as the size of the spanning tree grows. These results highlight the differences between both problems, as well as the importance of designing specific procedures for the 2SRNDC.

The increasing difference between the MST and the 2SRNDC solution also provides some explanation for the deterioration of the performance of the algorithm when the size of the instance grows. As the number of nodes grows, the scheduling part of the problem (the second stage) becomes more important, leading to a more intractable problem.

4.4. Analysis of results for the Chilean instance set

While the experiment with the randomly generated instances was geared towards understanding the behavior of the proposed solution method under varying conditions, the experiment with the Chilean instances tries to analyze the applicability of the method on a realistic situation. Note that the size of the Chilean instance is much larger than the randomly generated ones, thus showing the applicability of the proposed algorithm on larger-size instances, which feature the special characteristics of a real-life network.

In order to evaluate different situations, the restoration of the complete network as well as the reduced networks were consid-

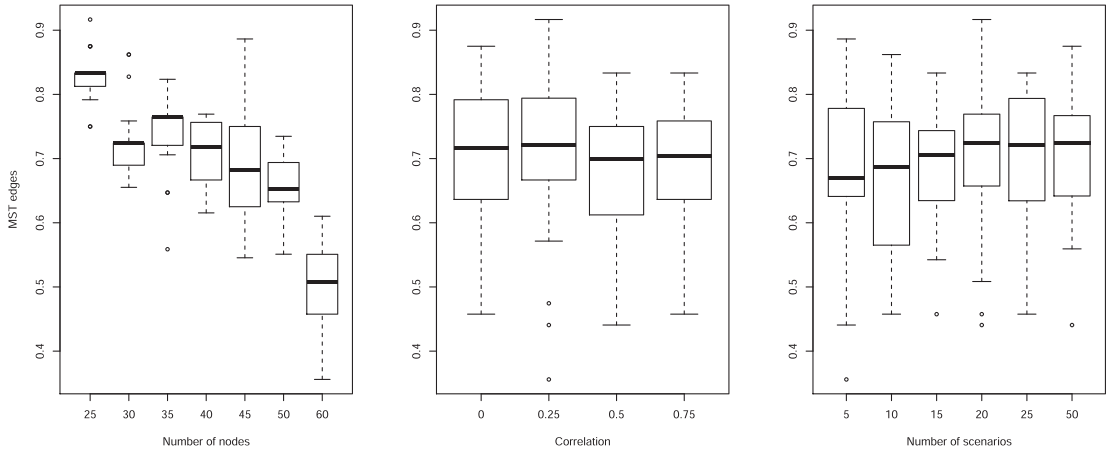


Fig. 5. Percentage of edges shared between the minimum spanning tree and the first stage solution. Instances are grouped according to their main characteristics (number of nodes, correlation between first and second stage costs, and number of scenarios.

Table 4
Detailed results for the Chilean instances grouped according to the aggregation parameter (0 km means no grouping) and the number of scenarios or the correlation. For each combination, the average gap is reported.

# Scenarios	0 km	5 km	10 km	15 km	20 km
5	14.669	6.971	2.890	0.423	0.000
10	15.652	8.013	2.667	0.782	0.000
15	15.432	8.190	2.990	0.917	0.000
20	15.890	8.194	3.254	0.924	0.000
25	16.089	8.375	3.355	1.061	0.000
50	16.374	8.296	3.536	1.342	0.000
ρ	0 km	5 km	10 km	15 km	20 km
0	16.977	9.043	3.940	1.540	0.000
0.25	17.044	8.755	3.496	1.140	0.000
0.5	15.924	8.114	3.086	0.728	0.000
0.75	12.793	6.113	1.939	0.225	0.000

ered. Table 4 provides the average optimality gaps for these instances when grouped according to the number of scenarios or the correlation factor.

The results show that the algorithm obtains smaller average gaps than those obtained for the randomly generated instances, even when the number of nodes is larger. Note that the smallest instances, grouping factor P=20 km, contain 91 nodes and they are solved to optimality for all of the scenarios and correlation cases. Even for the largest instances, those without aggregation or a lower level of aggregation, the results show that the optimality gaps are much smaller than the optimality gaps for smaller randomly generated instances. This is possibly as a result of the structure of the Chilean road network, which makes the problem more amenable than randomly generated instances. Furthermore, the results also show that larger correlation factors lead to instances with smaller optimality gaps, which further highlights the impact of more structured instances on the behavior of the algorithm.

5. Conclusions

In this paper we have considered a family of novel network design problems in which the network construction decisions are taken into account during the network design stage. These problems model situations in which: (i) the network is constructed during long periods of time; (ii) the nodes of the network receive service as soon as they are connected to the depot; and/or (iii) the network is subject to frequent reconstruction operations. More importantly, the proposed framework incorporates uncertainty as part of the construction phase. The resulting problem, which is referred to as the Two-Stage Robust Network Design and Construction Problem (2SRNDCP), considers the design of the network as the first-stage problem, while the second-stage problem optimizes a (robust) construction schedule of the first stage network. Among the possible formulations of both the first-stage and second-stage problems, we consider the construction of a spanning tree and the (worst-case) sum of connection times respectively.

These first-stage and second-stage problems were selected for their theoretical (they depict basic formulations of their respective problem categories) and practical interest (the first-stage problem tries to obtain a minimal network to reach any node of the network from a supply center, while the second-stage problem optimizes over the worst-case total connection time of all nodes to the depot). While the formulations can be seen as simplistic, the resulting problem is computationally hard, as even the second stage problem by itself is NP-hard [3].

In order to provide a solution for the problem, a constraint-generation approach is proposed. Furthermore, a matheuristic cut generator, derived from the optimal resolution of reduced instances of the second-stage problem, is incorporated. The proposed approach is compared with an extended formulation of the problem, as well as with the constraint-generation approach without the matheuristic cuts. The results show that incorporating the matheuristic significantly improves, in terms of lower optimality gaps, the solution provided by the basic constraint generation method.

The algorithm is tested on two different instance sets. The results for a randomly generated set of instances show that it is capable of optimally solving instances with up to 35 nodes, and it provides very reduced optimality gaps for instances with 40–45 nodes. The results for a set of instances derived from the Chilean road network show that much larger instances (approximately 100 nodes) can be solved to optimality due to the structured nature of the network.

According to the obtained results, we draw the following conclusions:

- While the resolution of an extended formulation of the problem is possible, its applicability is limited to very small instance sizes. For larger instances, the procedure based on dynamically adding constraints to bound the second-stage function (that uses a matheuristic method to generate additional (θ-C) constraints) outperforms other methods. The matheuristic relies on a branch-and-bound procedure to solve small-sized second-stage problems to optimality, and then derives cuts that

prune the solution space. The branch-and-bound method requires the use of some properties of the second-stage problem in order to speed up the search.

- Simultaneously considering the network construction (reconstruction) decisions and the network design decisions has a great impact on the implemented network. Even for moderately small instances ($n = 30$), there is a 30% difference on average among the edges of the minimum spanning tree (the optimal solution of the first stage), and the edges of the (optimal) 2SRNDCP solution. This difference, which increases along with the instance size, emphasizes the usefulness of the simultaneous resolution of the design and the construction problems.
- When compared with the related literature (see [3,4]), the proposed approach is effective for slightly smaller instances than those considered in the literature. Nonetheless, the problem addressed in this paper incorporates two additional levels of difficulty. On the one hand, we combine both design and scheduling decisions; and on the other hand, we consider the presence of uncertainty modeled by discrete scenarios and we tackle it via robust optimization. Consequently, the decrease in the size of the optimally solved instances can be attributed to these additional levels of complexity; furthermore, this does not hinder the ability of the algorithm to solve large-scale real-life instances.
- While this work considers a robust version of the problem with uncertainty in the second-stage costs, it is possible to apply the methodology to deterministic problems in which the realization of the costs of the construction phase is known in advance. Nevertheless, the authors believe that the importance of taking into account uncertainty in the construction phase balances out the additional computational burden that its treatment requires. Moreover, the results show that including a small number of scenarios is enough to protect the solution against uncertainty.

As a final remark, we highlight the versatility of the proposed solution scheme for other types of 2SRNDCP. Other formulations for the first or the second stage, like the Steiner tree or weighted completion time objectives, could be easily incorporated in the proposed framework after modifying their corresponding algorithmic components.

Acknowledgment

E. Álvarez-Miranda acknowledges the support of the Chilean Council of Scientific and Technological Research, CONICYT, through the grant FONDECYT N.11140060 and through the Complex Engineering Systems Institute (ICM-FIC:P05-004-F, CONICYT:FB0816).

Appendix A.

A.1. An extended formulation of 2SRSTSCT

Let A be the set of arcs of the bi-directed counterpart of $G = (V, E)$, $G_A = (V, A)$, such that $A = \{(ij), (ji) \mid e : \{i, j\} \in E\}$; likewise, $d_{ij} = d_{ji} = d_e$ and $c_{ij}^{\omega} = c_e^{\omega}$, $\forall e$: $\{i, j\} \in E$.

Let $\mathbf{x} \in \{0, 1\}^{|A|}$ be a binary vector such that $x_{ij} = 1$, if arc $(ij) \in A$ belongs to a spanning arborescence of G_A and $x_{ij} = 0$ otherwise. Moreover, let $\mathbf{y} = \left(\mathbf{y}^1, \ldots, \mathbf{y}^{\omega}, \ldots, \mathbf{y}^{|\Omega|}\right)$ be a collection of binary vectors $\mathbf{y}^{\omega} \in \{0, 1\}^{|A| \times (|V|-1)}$, such that for each $\omega \in \Omega$ and $k = \{1, \ldots, n-1\}$, $y_{ij}^{k\omega} = 1$ if arc $e \in E$ is the kth constructed arc if scenario $\omega \in \Omega$ is realized, and $y_e^{k\omega} = 0$ otherwise.

We will use the following additional notation: a set of vertices $S \subseteq V$ $(S \neq \emptyset)$ and its complement $\bar{S} = V \backslash S$, induce two directed cuts, $\delta^+(S) = \left\{(ij) \mid i \in S, j \in \bar{S}\right\}$ and $\delta^-(S) = \left\{(ij) \mid i \in \bar{S}, j \in S\right\}$.

Then formulation (2SRSTC.1)–(2SRSTC.11) constitute a valid formulation for the Two-Stage Robust Spanning Tree Construction Problem (2SRSTC).

$$\min \sum_{(ij)\in A} d_{ij}x_{ij} + \theta \tag{2SRSTC.1}$$

$$\text{s.t.} \quad \sum_{(ij)\in\delta^-(S)} x_{ij} \geq 1, \ \forall S \subseteq V \backslash \{r\} \ S \neq \emptyset \tag{2SRSTC.2}$$

$$\sum_{(ij)\in\delta^-(j)} x_{ij} = 1, \ \forall j \in V \setminus \{r\}. \tag{2SRSTC.3}$$

$$\sum_{(rj)\in\delta^+(r)} y_{rj}^{1\omega} = 1, \ \forall \omega \in \Omega \tag{2SRSTC.4}$$

$$\sum_{(ij)\in A} y_{ij}^{k} = 1, \ \forall k \in \{2, \ldots, |V|-1\} \ \forall \omega \in \Omega \tag{2SRSTC.5}$$

$$\sum_{k=1}^{n-1} \sum_{(ij)\in\delta^-(j)} y_{ij}^{k\omega} = 1, \ \forall j \in V \setminus \{r\} \ \forall \omega \in \Omega \tag{2SRSTC.6}$$

$$y_{ij}^{1\omega} = 0, \ \forall (ij) \in A \mid_{i \neq r} \ \forall \omega \in \Omega \tag{2SRSTC.7}$$

$$y_{ij}^{k\omega} \leq x_{ij}, \ \forall (ij) \in A \ \forall \omega \in \Omega \tag{2SRSTC.8}$$

$$\theta \leq \sum_{k=1}^{|V|-1} \sum_{(ij)\in A} (|V| - k) c_{ij}^{\omega} y_{ij}^{k\omega}, \ \forall \omega \in \Omega \tag{2SRSTC.9}$$

$$x_{ij} \in \{0, 1\}, \ \forall (i, j) \in A \tag{2SRSTC.10}$$

$$y_{ij}^{k\omega} \in \{0, 1\}, \ \forall (i, j) \in A \ \omega \in \Omega \ k \in \{1, \ldots, |V|-1\}. \tag{2SRSTC.11}$$

Constraints sets (2SRSTC.2), (2SRSTC.3) and (2SRSTC.10) define the first stage problem. Constraint (2SRSTC.2), which are exponential in number, are known as *cut-set* or *connectivity* inequalities and they ensure that there is a directed path from the root r to each other node $v \in V \backslash \{r\}$. This type of constraints is usually used in the context of effective branch-and-cut procedures and its separation can be performed in polynomial time using a maximum-flow algorithm on a *support graph* with arc-capacities given by the current fractional solution $\tilde{\mathbf{x}}$. Constraints (2SRSTC.3), commonly referred as *in-degree* constraints, ensure the solution to be cycle-free.

Constraint sets (2SRSTC.4)–(2SRSTC.8) as well as the domain definition, constraint set (2SRSTC.11) define the extended form of the second stage. Constraint (2SRSTC.4) imposes that the first constructed arc starts at r. The fact that each of the arcs are constructed only once is modeled by (2SRSTC.5). For each scenario constraint (2SRSTC.6) models that every node $j \in V \backslash \{r\}$ must be reached by exactly one constructed arc $(ij) \in \delta^-(j)$. The connectivity of the construction scheduling is imposed by (2SRSTC.7); it imposes that if an arc $(ij) \in A|_{i \neq r}$ is the kth constructed arc, then at least one arc incident to i ($(hi) \in \delta^-(i)$) must have been previously constructed. Constraint (2SRSTC.8) relates variables $\mathbf{y}^{\omega}$ with the solution $\mathbf{x}$ in such a way that an arc $(ij) \in A$ can be constructed ($y_{ij}^{k\omega} = 1$ for some $k = \{1, \ldots, n-1\}$) if and only if it has been chosen as part of the designed network ($x_{ij} = 1$). The value of θ is obtained in (2SRSTC.9). Finally, (2SRSTC.11) define the domain of the variables of the extended form of the second stage.

Table 5
Detailed results with instances grouped according to size and number of scenarios.

# Nodes	# Scenarios	# Cut-sets	# (θ-C)	# Matheuristic (θ-C)	# BCnodes	# Opt	Av. gap	Av. time
25	5	24.5	64.2	32.0	119.0	4	0.0	4.2
	10	38.0	126.2	101.2	186.8	4	0.0	14.8
	15	35.0	254.5	159.2	245.8	4	0.0	21.8
	20	38.5	373.2	231.8	297.8	4	0.0	53.8
	25	43.5	278.5	263.2	229.8	4	0.0	50.3
	50	41.5	729.5	396.2	247.2	4	0.0	105.7
30	5	42.0	146.5	75.8	371.8	4	0.0	12.6
	10	47.0	310.2	170.5	427.5	4	0.0	18.8
	15	54.0	500.8	289.5	595.2	4	0.0	27.3
	20	55.0	634.8	529.0	663.0	4	0.0	58.8
	25	52.5	912.0	484.0	745.8	4	0.0	87.6
	50	52.0	1588.5	848.5	693.5	4	0.0	245.2
35	5	80.5	401.2	263.0	2052.5	4	0.0	86.3
	10	120.0	1127.0	644.0	4111.2	4	0.0	282.9
	15	125.0	1406.0	783.2	3195.5	4	0.0	248.9
	20	108.0	1594.8	959.8	2652.5	4	0.0	293.5
	25	116.5	1871.8	1506.8	3916.8	4	0.0	542.5
	50	137.0	4706.2	3223.2	4626.8	3	0.3	1125.8
40	5	180.5	2436.0	2262.5	25,864.8	4	0.0	1597.0
	10	183.5	4023.0	3424.0	18,056.5	2	1.0	1976.4
	15	183.5	4732.2	4375.0	15,059.5	2	1.0	2180.8
	20	183.0	6304.2	5438.2	12,023.5	0	1.7	2400.0
	25	171.0	6530.2	5683.5	10,463.2	0	1.8	2400.0
	50	156.5	9244.5	7031.5	6000.5	0	2.9	2400.0
50	5	209.5	2046.5	2811.2	17,428.0	0	6.0	2400.0
	10	217.5	2767.0	4058.2	11,600.2	0	7.0	2400.0
	15	213.5	3687.8	4673.5	9016.8	0	7.9	2400.0
	20	192.0	3843.5	5424.8	7683.0	0	7.8	2400.0
	25	190.0	4360.0	5923.5	6630.5	0	8.4	2400.0
	50	168.0	5735.2	8246.2	4562.8	0	8.9	2400.0
60	5	318.5	1472.2	2412.2	14,207.5	0	10.7	2400.0
	10	308.0	1934.2	3303.8	9530.0	0	11.9	2400.0
	15	297.0	2468.2	3996.0	8460.5	0	11.6	2400.0
	20	284.5	3115.5	5092.0	6874.5	0	12.9	2400.0
	25	281.0	3267.0	5229.0	5760.5	0	13.3	2400.0
	50	261.0	4434.5	7118.2	3899.0	0	14.1	2400.0
75	5	624.0	1069.0	2019.0	11,346.2	0	17.9	2400.0
	10	610.0	1687.8	2820.5	8272.8	0	19.1	2400.0
	15	550.5	1738.2	3329.5	6748.0	0	19.4	2400.0
	20	559.5	2118.8	3851.2	5756.0	0	19.6	2400.0
	25	525.5	2635.2	4473.2	5590.0	0	19.8	2400.0
	50	478.0	3435.0	6061.0	3522.8	0	21.2	2400.0
100	5	1181.5	603.5	1518.8	10,513.2	0	27.3	2400.0
	10	869.0	910.0	2421.5	7220.5	0	28.9	2400.0
	15	923.5	1180.8	2888.5	6340.8	0	29.3	2400.0
	20	860.0	1385.8	3006.2	5315.2	0	29.3	2400.0
	25	880.5	1469.5	3996.5	5058.0	0	30.0	2400.0
	50	750.5	2175.5	5094.5	3421.5	0	31.6	2400.0

A.2. Detailed results

Table 5 provides an analysis of the results grouped by instance size and number of scenarios (4 instances per group), while Table 6 provides the same results grouped by instance size and correlation between the first-stage and the second-stage costs (6 instances per group).

Table 5 shows an increase in the required running time as the number of nodes and scenarios grow. Note that for instances with $n > 50$, the algorithm fails to find any optimal solution and thus the average running time equals the time limit of the algorithm. Table 5 also shows that the number of nodes and cuts dramatically increases for $n \geq 40$, showing the limits of the proposed approach and the difficulty of the problem.

A.3. Graphical representation of the Chilean network

Fig. 6(a) shows a representation of the complete road network. A detail of the network corresponding to the central area of Chile (where more than 75% of the population is concentrated) is displayed in Fig. 6(b).

Fig. 6(c) and (d) show the resulting instances when considering $P = 10$ and $P = 20$ respectively.

Table 6
Detailed results with instances grouped according to size and correlation.

# Nodes	Correlation	# Cut-sets	# (θ-C)	# Matheuristic (θ-C)	# BCnodes	# Opt	Av. gap	Av. time
25	0	31.0	166.7	133.5	149.2	6	0.0	34.3
	25	35.3	228.7	167.0	207.5	6	0.0	71.5
	50	42.3	356.8	240.3	280.0	6	0.0	24.2
	75	38.7	465.3	248.3	247.5	6	0.0	37.2
30	0	45.3	594.3	302.8	557.0	6	0.0	66.8
	25	52.7	654.2	346.7	639.7	6	0.0	92.5
	50	50.0	686.3	465.5	576.7	6	0.0	99.7
	75	53.7	793.7	483.2	557.8	6	0.0	41.2
35	0	98.7	1231.2	679.7	2038.7	6	0.0	149.2
	25	112.0	1603.3	1122.3	3180.5	6	0.0	319.7
	50	125.0	2306.2	1618.5	4731.5	5	0.2	863.1
	75	122.3	2264.0	1499.5	3752.8	6	0.0	387.9
40	0	161.7	5131.0	3959.3	13,343.7	3	0.7	1863.7
	25	176.0	5122.7	4350.8	16,808.5	3	0.8	2079.3
	50	180.3	5572.5	5015.7	14,769.3	1	1.8	2329.8
	75	187.3	6354.0	5484.0	13,390.5	1	2.2	2363.3
45	0	131.0	4266.2	6350.3	10,526.0	0	4.2	2400.0
	25	142.3	3972.7	6506.8	10,820.3	0	5.1	2400.0
	50	149.3	3608.2	6206.5	9494.2	0	6.8	2400.0
	75	140.3	3748.5	5551.0	8539.8	0	6.9	2400.0
50	0	195.0	3819.2	5054.5	9164.7	0	7.2	2400.0
	25	201.3	3862.8	5258.3	10,126.7	0	7.5	2400.0
	50	194.7	3553.3	5197.0	9216.5	0	7.9	2400.0
	75	202.7	3724.7	5248.5	9439.7	0	8.1	2400.0
60	0	271.3	2855.2	4467.5	8526.3	0	11.2	2400.0
	25	290.7	2940.5	4851.8	8435.0	0	12.0	2400.0
	50	303.7	2646.3	4385.8	7762.8	0	13.3	2400.0
	75	301.0	2685.8	4395.7	7763.8	0	13.1	2400.0
75	0	545.0	2047.8	3444.2	6533.2	0	18.0	2400.0
	25	555.3	2295.8	3870.3	7255.3	0	19.6	2400.0
	50	576.7	2091.8	3775.3	6966.7	0	20.0	2400.0
	75	554.7	2020.5	3946.5	6735.3	0	20.3	2400.0
100	0	912.7	1285.8	3020.8	6279.7	0	28.0	2400.0
	25	917.0	1216.7	3147.3	5909.0	0	29.6	2400.0
	50	927.3	1428.0	3265.0	6537.8	0	30.5	2400.0
	75	886.3	1219.5	3184.2	6519.7	0	29.4	2400.0

(a) `CHILE` Instance (b) Detail of the central Area (c) 10km-`CHILE` (d) 20km-`CHILE`

Fig. 6. Examples of the underlying network of the instances considered for computation.

References

[1] Álvarez-Miranda E, Ljubić I, Raghavan S, Toth P. The recoverable robust two-level network design problem. INFORMS J Comput 2014;27:1–19.
[2] Atamtürk A, Zhang M. Two-stage robust network flow and design under demand uncertainty. Oper Res 2007;55(4):662–73.
[3] Averbakh I, Pereira J. The flowtime network construction problem. IIE Trans 2012;44(8):681–94.
[4] Averbakh I, Pereira J. Network construction problems with due dates. Eur J Oper Res 2015;244(3):715–29.
[5] Baxter M, Elgindy T, Ernst A, Kalinowski T, Savelsbergh M. Incremental network design with shortest paths. Eur J Oper Res 2014;238(3):675–84.
[6] Ben-Tal A, Goryashko A, Guslitzer E, Nemirovski A. Adjustable robust solutions of uncertain linear programs. Math Program Ser A 2004;99:351–76.
[7] Ben-Tal, A, El-Ghaoui, L Nemirovski, A (Eds.) (2010). Robust optimization. Series in applied mathematics, Princeton. 1st ed.
[8] Bertsimas D, Sim M. The price of robustness. Oper Res 2004;52(1):35–53.
[9] Birge J, Louveaux F. Series in operations research and financial engineering. Introduction to stochastic programming. Springer; 2011.
[10] Brucker, P, (2007). Scheduling algorithms, 5th ed. Springer.
[11] Büsing C. Recoverable robust shortest path problems. Networks 2012;59(1):181–9.
[12] Emmons H. One-machine sequencing to minimize certain functions of job tardiness. Oper Res 1969;17(4):701–15.
[13] Engel, K., Kalinowski, T., & Savelsbergh, M. (2013). Incremental network design with minimum spanning trees. URL http://arxiv.org/abs/1306.1926.
[14] Johnson D, Minkoff M, Phillips S. The prize collecting steiner tree problem: theory and practice. In: Proceedings of the 11th symposium on discrete algorithms; 2000. p. 760–9. ACM/SIAM
[15] Kalinowski, T., Matsypura, D., & Savelsbergh, M. (2014). Incremental network design with maximum flows. URL http://www.sciencedirect.com/science/article/pii/S0377221714008078.
[16] Koch T, Martin A. Solving Steiner tree problems in graphs to optimality. Networks 1998;32(3):207–32.
[17] Kouvelis P, Yu G. Robust discrete optimization and its applications. Nonconvex optimization and its applications. Kluwer Academic Publishers; 1997. 1st ed.
[18] Laporte G, Louveaux F. The integer l-shaped method for stochastic integer programs with complete recourse. Oper Res Lett 1993;13(3):133–42.
[19] Montgomery D. Design and analysis of experiments. Wiley; 2012. 8th ed.
[20] Nurre S, Sharkey T. Integrated network design and scheduling problems with parallel identical machines: complexity results and dispatching rules. Networks 2014;63(4):306–26.
[21] Nurre S, Cavdaroglu B, Mitchell J, Sharkey T, Wallace W. Restoring infrastructure systems: an integrated network design and scheduling (inds) problem. Eur J Oper Res 2012;223(3):794–806.
[22] Quilliot A. Network design problems: fundamental methods. In: Paschos V, editor. Applications of combinatorial optimization. John Wiley & Sons; 2013. p. 253–89. Chapter 9.
[23] Zhao L, Zeng B. An exact algorithm for two-stage robust optimization with mixed integer recourse problems. Technical Report; 2012.