Received: 8 August 2018 | Revised: 28 October 2019 | Accepted: 3 November 2019
DOI: 10.1002/net.21922

RESEARCH ARTICLE

WILEY

Two-stage stochastic minimum $s-t$ cut problems: Formulations, complexity and decomposition algorithms

Steffen Rebennack[1] | Oleg A. Prokopyev[2] | Bismark Singh[1,3]

[1] Department of Stochastic Optimization, Institute for Operations Research, Karlsruhe Institute of Technology, Karlsruhe, Germany
[2] Department of Industrial Engineering, University of Pittsburgh, Pittsburgh, Pennsylvania,
[3] Discrete Mathematics, Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany

Correspondence
Steffen Rebennack, Institute for Operations Research, Karlsruhe Institute of Technology, Karlsruhe, Germany.
Email: steffen.rebennack@kit.edu

Funding information
This research was supported by the U.S. Air Force Office of Scientific Research, FA9550-08-1-0268 and FA9550-11-1-0037.

Abstract

We introduce the two-stage stochastic minimum $s-t$ cut problem. Based on a classical linear 0-1 programming model for the deterministic minimum $s-t$ cut problem, we provide a mathematical programming formulation for the proposed stochastic extension. We show that its constraint matrix loses the total unimodularity property, however, preserves it if the considered graph is a tree. This fact turns out to be not surprising as we prove that the considered problem is $\mathcal{NP}$-hard in general, but admits a linear time solution algorithm when the graph is a tree. We exploit the special structure of the problem and propose a tailored Benders decomposition algorithm. We evaluate the computational efficiency of this algorithm by solving the Benders dual subproblems as max-flow problems. For many tested instances, we outperform a standard Benders decomposition by two orders of magnitude with the Benders decomposition exploiting the max-flow structure of the subproblems.

KEYWORDS

Benders decomposition, combinatorial optimization, complexity, minimum $s-t$ cut problem, total unimodularity, two-stage stochastic programming

1 | INTRODUCTION

Let $G = (V, A)$ be a directed graph with node set V, arc set $A \subseteq V \times V$, and nonnegative costs c_{ij} given for each arc $ij \in A$. The *minimum $s-t$ cut problem* for directed graphs can be defined as follows. For a given directed connected graph $G = (V, A)$ with root node s and terminal node t, the task is to find a node set $S \subset V$ with $s \in S$ and a node set $T \subset V$ with $t \in T$, such that $S \cup T = V$, $S \cap T = \emptyset$ and the *cost of the cut* $c[S, T] := \sum_{ij \in A : i \in S \wedge j \in T} c_{ij}$ *is minimized*.

The minimum $s-t$ cut problem and the max-flow problem are dual problems to each other. This relation is called the *max-flow min-cut theorem* and was first proven in [11]. This duality enhanced the development of many polynomial time algorithms for computing minimum $s-t$ cuts; see, for example, [2].

The following mathematical programming formulation of the minimum $s-t$ cut problem dates back to Ford and Fulkerson [12]. Let

$$x_i = \begin{cases} 1, & \text{if } i \in T \\ 0, & \text{if } i \in S \end{cases} \quad \forall i \in V \quad \text{and} \quad y_{ij} = \begin{cases} 1, & \text{if } i \in S,\ j \in T \\ 0, & \text{otherwise.} \end{cases} \quad \forall ij \in A. \tag{1.1}$$

This allows the following linear 0-1 programming formulation for the minimum $s-t$ cut problem:

$$\min_{x,y} \sum_{ij \in A} c_{ij} y_{ij} \tag{1.2a}$$

$$\text{s.t.} \quad y_{ij} \geq x_j - x_i, \qquad \forall ij \in A; \tag{1.2b}$$

$$x_s = 0; \;\; x_t = 1; \tag{1.2c}$$

$$x_i \in \{0,1\}, y_{ij} \in \{0,1\}, \qquad \forall i \in V, \quad \forall ij \in A. \tag{1.2d}$$

Note that Ford and Fulkerson consider the equation $x_t - x_s = 1$ in [12] instead of Equation (1.2c). The constraint matrix defined by (1.2b)–(1.2c) is totally unimodular (TU) [12] (see further discussion in Section 2.2), allowing the relaxation of the variables x and y to be nonnegative, continuous, and bounded above by 1. This provides another indication of the fact that the minimum $s-t$ cut problem is polynomially solvable.

An alternative definition of the minimum $s-t$ cut problem can be provided as follows. Define a *cutset* as a set of arcs whose removal ensures that there is no directed path from s to t. Then the minimum $s-t$ cut problem can be defined as the problem of finding a cutset of minimum weight. Note that model (1.2) has a clear interpretation in this framework since variable y_{ij} is 1 if the corresponding arc ij belongs to the required cutset.

This interpretation provides intuitive practical motivation for the minimum $s-t$ cut problem in the military and law-enforcement contexts. Assume that G is an adversarial transportation (or communication) network and the decision-maker is interested in disrupting the connectivity between two predefined nodes (i.e., s and t) in the network. Then, each arc cost can be interpreted as the amount of the interdiction resources necessary for the decision-maker to terminate direct communication (i.e., an arc) between the corresponding node pair. Consequently, an optimal solution of the minimum $s-t$ cut problem provides an optimal "attack" and the minimum amount of the resources required for the decision-maker to completely disconnect the $s-t$ node pair. Clearly, this interpretation can be generalized to capture network reliability and survivability considerations, where arcs can fail due to some natural reasons; see discussions and examples in [7, 20, 21]. Furthermore, it should be noted that in the transportation setting an $s-t$ cutset of minimum weight can be viewed as a "bottleneck" limiting the maximum flow through the network. In fact, military considerations (in particular, underlying interdiction ideas) were one of the original motivations for studying the minimum cut problem in the literature [29]. The now de-classified Harris-Ross report [17] was one of the earliest related studies, where the authors attempted to identify the bottleneck of the railway network in the Western Soviet Union and Eastern Europe. Finally, we refer the reader to [2, 19, 25] and the references therein for more detailed discussions of the minimum cut problem, its applications, and solution approaches.

Stochastic programming models exploit the fact that in many real-life applications probability distributions governing the problem data are known or can be estimated in advance. Numerous studies over the past several decades demonstrate potential benefits of stochastic programming models over their deterministic counterparts [28, 36]. In particular, the related literature contains a number of studies that consider stochastic extensions of various classical graph and network design problems. Examples include two-stage stochastic extensions of maximum weight matching [23], shortest path [15], minimum spanning tree [10, 13], and Steiner tree [16] problems. For an introduction to stochastic programming, we refer the reader to [4].

In the aforementioned military and law-enforcement application contexts, the decision-maker often does not have full information about the underlying network. Thus, it is natural to consider settings with multiple decision-making epochs—two in the current study—where the decision-maker does not need to make all of his/her decisions in the same time period, but instead has an opportunity to construct a valid cutset over two decision-making epochs. Specifically, the decision-maker can disrupt some arcs in the network initially (i.e., in the first stage). The uncertainty in the second stage (i.e., the second decision-making epoch) is characterized by some known probability distribution (e.g., estimated based on the available intelligence and/or surveillance information) and the decision-maker can disrupt distinct arc subsets in the second stage depending on the particular scenario, realized with the overall goal to construct a cutset of the minimum expected cost. Therefore, the decision-maker does not need to commit to all of his/her decisions right away (i.e., in the first stage) but instead attempts to take advantage of his/her knowledge of the probability distributions governing the problem data (i.e., arc costs and specific terminal node for each scenario) in the future.

Motivated by the discussion above, we introduce the following two-stage stochastic extension of the original deterministic problem.

Definition 1. Given a directed graph $G = (V, A)$ with node set V, arc set $A \subseteq V \times V$, a root node $s \in$ V, and K scenarios. The kth scenario consists of a single terminal node t^k and has a probability p^k, where $0 < p^k < 1$, of being realized. Arc $ij \in A$ has nonnegative cost c_{ij} in the first stage and nonnegative cost d_{ij}^k in the recourse stage (or, second stage) if the kth scenario is realized. The task is to find a set of arcs E_0 to cut in the first stage, and for each scenario k, an arc set E_k to be cut in the recourse stage if scenario k is realized, such that removing $E_0 \cup E_k$ from the graph G disconnects s from the terminal t^k. The objective is to minimize the total expected cost; that is, min $\left(\sum_{ij \in E_0} c_{ij} + \sum_{k=1}^{K} p^k \sum_{ij \in E_k} d_{ij}^k\right)$.

Previous studies have considered *stochastic* min-cut problems in various forms. In [18], the authors study a network with random arc capacities following a discrete probability distribution. There is also work on finding the min-cut in a network where arcs fail randomly [5, 33]. We instead consider the min-cut problem of disconnecting a root node from a terminal node, where the terminal node and the arc capacities are random. In [14], the authors present an alternate formulation, which provides an approximate solution, to this problem. The authors in [9, 15] discuss a somewhat related *robust $s-t$ min-cut problem*, where the task is to minimize the *maximum* cost over all scenarios while disconnecting s from terminals t^k. This robust min-cut problem is known to be APX-hard [22]. The robust problem differs from our stochastic problem, as in the latter problem we identify a min-cut for every scenario.

In our problem statement (see Definition 1) we make three assumptions. First, we assume that the uncertainty in the second stage is characterized by a finite number of scenarios. This assumption is standard in the related stochastic mixed integer and combinatorial optimization literature; see, for example, [30] for the related discussion on the stability issues. For problems with a prohibitively large (or infinitely many) number of scenarios, we assume that the problem given in Definition 1 is obtained using the sample average approximation method. Our second assumption is that the root node is known to the decision-maker and only the terminal nodes are uncertain. In making this assumption we follow the aforementioned related studies in [9, 15]; furthermore, this assumption simplifies our technical analysis in this paper. In the military and law-enforcement settings used as our motivating application context, it implies that whenever the decision-maker is interested in disconnecting a particular node pair in the adversarial network, the decision-maker has complete knowledge about at least one node from the pair (e.g., a source of smuggling operations but not the destination which may depend on the scenario realized). Relaxing this assumption provides an interesting avenue for future research. Our third assumption is that the decision-maker is risk-neutral as only the expected cost is optimized. Naturally, one can consider more general risk-averse versions of the problem, for example, by adding some dispersion statistic into the objective function with an appropriate nonnegative multiplier to capture risk considerations [1].

This article has the following contributions:

(i) We extend the classical $s-t$ min-cut problem to a stochastic two-stage $s-t$ min-cut problem, and provide a mixed-integer linear programming (MILP) formulation for the latter.

(ii) We show that total unimodularity is preserved if the graph is a tree and we prove $\mathcal{NP}$-hardness for general graphs.

(iii) We propose a tailored Benders decomposition algorithm with an efficient implementation where Ford-and-Fulkerson's max-flow algorithm is used to solve the Benders subproblems. The proposed Benders decomposition algorithm uses the classical Benders optimality cuts in the context of two-stage stochastic optimization. With that, it is a direct application of the well-known L-shaped method with some computational enhancements.

The remainder of this paper is organized as follows. In Section 2, we provide a linear mixed 0-1 programming formulation of the two-stage stochastic minimum $s-t$ cut problem that is a generalization of the classical model (1.2). Unfortunately, the constraint matrix of the proposed mathematical program loses the total unimodularity property (Section 2.2) of the original deterministic formulation; however, this property is preserved if graph G is a tree (Section 2.3). This fact turns out to be not surprising as we prove in Section 3 that the considered problem is $\mathcal{NP}$-hard, while a linear time solution algorithm is available when the graph is a tree (see Section 4). Our tailored Benders decomposition algorithm is presented in Section 5 followed by computational results in Section 6. Finally, Section 7 concludes the discussion.

2 | MATHEMATICAL PROGRAMMING FORMULATION

We discuss the two-stage stochastic minimum $s-t$ cut problem considering the graph in Figure 1. Two scenarios are given with equal probabilities of 0.5. This graph has four nodes with node 1 as the root and node 4 as the terminal node for both scenarios.

In the two-stage stochastic minimum $s-t$ cut problem, one has to decide which arcs are cut in the first and second stage, where the cut in the second stage depends on the particular second-stage scenario. An optimal solution using this arc-based interpretation is shown in Figure 1B. In the first stage, both arcs (2,3) and (3,2) are a cut. In the second stage, arcs (1,3) and (2,4) are a cut for scenario 1, while arcs (1,2) and (3,4) are a cut for scenario two. This way, the optimal objective function value is 4.

Interestingly, in both scenarios, the resulting graph is disconnected after removing the arcs but the removal of the arcs does not correspond to a "minimum cut" in the classic sense; that is, as a partition of the nodes. This is the case, since in both scenarios arcs (2,3) and (3,2) are cut in the first stage. We interpret this solution as *hedging* of arcs. Specifically, note that the resulting arcs define a valid cutset. On the other hand, let us consider each scenario independently after the arcs that are cut in the first stage are removed from G. Then we can observe that the resulting subproblem for each scenario is a classical minimum $s-t$ cut problem that can be equivalently interpreted either as a partition of the nodes or a cutset of arcs. On the other hand, for each scenario one of the arcs removed in the first stage is not actually needed to form a valid cut. In other words, in the first

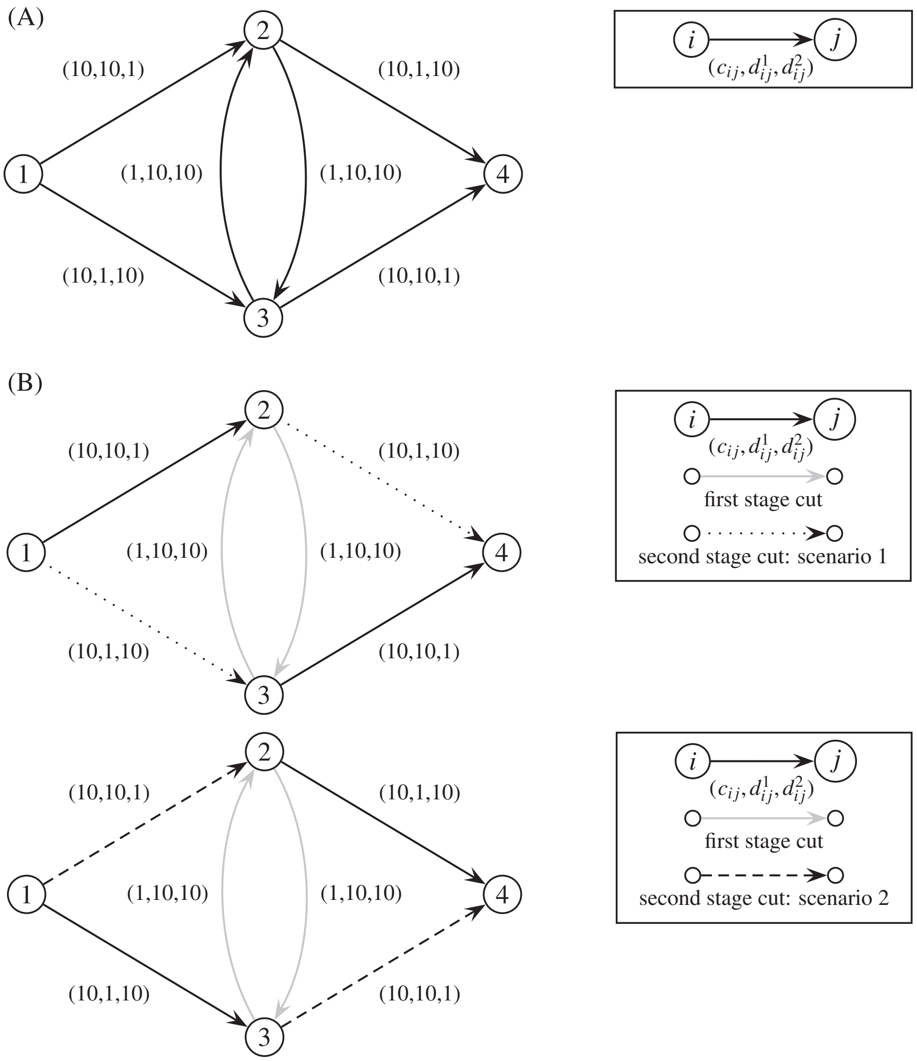
(A)
(B)
1
2
3
4
(10,10,1)
(10,1,10)
(1,10,10)
(1,10,10)
(10,1,10)
(10,10,1)
i
j
$(c_{ij}, d_{ij}^1, d_{ij}^2)$
first stage cut
second stage cut: scenario 1
second stage cut: scenario 2

FIGURE 1 Example of a two-stage minimum $s-t$ cut instance. (A) Node 1 is the root node and node 4 is the terminal node. Given are two scenarios with equal probability. (B) Hedging: in the first stage, both arcs (2,3) and (3,2) are cut. The cost of this minimum cut is 4. The top graph shows the optimal cuts for scenario 1 and the bottom graph for scenario 2 (together with the first-stage cuts)

stage the decision-maker removes an extra arc (with respect to each scenario) in order to *hedge* his/her exposure to the arc costs in the second stage.

We now discuss two cases when a specific structure on the costs of arcs is present.

Case 1: $c_{ij} \leq p^k d_{ij}^k,\ \forall ij \in A,\ k = 1, \dots, K$. The first-stage cost for each arc is less than or equal to the second-stage costs in each of the K scenarios, weighted with the scenario probability. Thus, there is no need to cut in the second stage. Now, the problem transforms into a cut problem, where K terminals have to be cut from a single source s. This problem can be transformed into a (deterministic) minimum $s-t$ cut problem by introducing a super terminal node t which is connected to each of the K terminals with arc cost $+\infty$.

Case 2: $c_{ij} \geq d_{ij}^k,\ \forall ij \in A,\ k = 1, \dots, K$. The first-stage arc cost is greater than or equal to the cost of cutting in the second stage at any of the scenarios. In this case, no arcs need to be cut in the first stage and the second-stage problem decomposes into K independent (deterministic) minimum $s-t$ cut problems.

Thus, in both of the above cases, the two-stage stochastic minimum $s-t$ cut problem is solvable in polynomial time. This is summarized in the following proposition.

Proposition 1. *The two-stage stochastic minimum $s-t$ cut problem is solvable in strongly polynomial time for arbitrary graphs if one of the following two cases holds*: $c_{ij} \leq \min_{k=1,\dots,K} p^k d_{ij}^k,\ \forall ij \in A$, *or* $c_{ij} \geq \max_{k=1,\dots,K} d_{ij}^k,\ \forall ij \in A$.

Note that the conditions presented in Cases 1 and 2 can be exploited in a preprocessing step. For example, if for some arc $ij \in A$, $c_{ij} \leq p^k d_{ij}^k$ for all $k = 1, \dots, K$, then arc ij does not need to be cut in the second stage for any scenario, which reduces the problem size (in terms of decision variables).

Allowing “hedging” of arcs, we obtain the following formulation for the two-stage stochastic minimum $s-t$ cut problem

$$z^* = \min \quad \sum_{ij\in A} c_{ij}y_{ij} + \sum_{k=1}^{K} p^k \sum_{ij\in A} d_{ij}^k u_{ij}^k \tag{2.1a}$$

$$\text{s.t.} \quad u_{ij}^k + y_{ij} \geq x_j^k - x_i^k, \qquad \forall ij \in A, \quad k = 1, \dots, K; \tag{2.1b}$$

$$x_s^k = 0, \quad k = 1, \dots, K; \tag{2.1c}$$

$$x_{t^k}^k = 1, \quad k = 1, \dots, K; \tag{2.1d}$$

$$x_i^k, y_{ij}, u_{ij}^k \in \{0, 1\}, \qquad \forall i \in V, \quad \forall ij \in A, \quad k = 1, \dots, K, \tag{2.1e}$$

where we define variables u_{ij}^k similar to Equation (1.1) as the arc to be cut for scenario k; variable x_i^k has value one if node $i \in V$ belongs to the sink set T^k, otherwise it has value 0 and belongs to the source set S^k; that is, $T^k \cup S^k = V$ and $T^k \cap S^k = \emptyset$.

Proposition 2. *Model* (2.1) *formulates the two-stage stochastic minimum $s-t$ cut problem correctly.*

Proof. Let arc sets E_0 and E_k define a two-stage $s-t$ cut for graph G. Arc set E_k defines sets S^k and T^k for each scenario k. With this, assign variables x^k values either 0 or 1 accordingly. The cut variables y and u^k obtain their values according to the arc sets E_0 and E_k. With this assignment, inequality (2.1b) is satisfied because otherwise, the sets S^k and T^k do not define a cut. The objective function value of this cut is calculated correctly.

It remains to show that any feasible solution of model (2.1) defines a two-stage $s-t$ cut for graph G. Therefore, assume that we are given a feasible solution of model (2.1) and that the graph is not disconnected. This implies that, for at least one scenario k, there is a (directed) path from root s to terminal t^k. Inequality (2.1b) implies that all variables x_i^k for nodes i along this path have the same value, which contradicts Equations (2.1c) and (2.1d). ■

The next proposition states that it is never optimal to cut the same arc both in the first and the second stage, whenever the arc costs are nonzero.

Proposition 3. *The inequality*

$$u_{ij}^k + y_{ij} \leq 1, \qquad \forall ij \in A, \;\; k = 1, \dots, K, \tag{2.2}$$

strengthens model (2.1). *If $d_{ij}^k > 0$, then inequality* (2.2) *does not cut away any optimal solution of model* (2.1).

Proof. Clearly the inequality is valid if at least one of the two variables y_{ij} or u_{ij}^k is not 1. Therefore, assume that there exists an arc $ij \in A$ and a scenario index $k \in \{1, \dots, K\}$, such that $u_{ij}^k = y_{ij} = 1$ in an optimal solution with objective function value z^*. By assigning $u_{ij}^k = 0$ and leaving all other decision variables unchanged, the objective function reduces by $p_k d_{ij}^k$. This contradicts optimality, unless $d_{ij}^k = 0$. ■

Recognize that model (2.1) does not involve any variables x_i for the first stage, but only for the recourse stage. Furthermore, variables y and u^k can be relaxed to be nonnegative continuous if variables x^k are binary. In order to see this, consider an optimal, fractional solution for variables y and u^k, as well as the corresponding binary solution values of variables x_i^k and x_j^k, and assume that all $d_{ij}^k > 0$. This implies that there must be an arc $ij \in A$ with $y_{ij} = l \in (0, 1)$; otherwise, each fractional variable u_{ij}^k implies that $d_{ij}^k = 0$. Then, inequality (2.1b) implies that either $u_{ij}^k = 1 - l$ if $x_j^k - x_i^k = 1$ or $u_{ij}^k = 0$ otherwise. Considering all scenarios k, arc $ij \in A$ contributes to the objective function the value $c_{ij} \cdot l + \sum_{k=1}^{K} p^k d_{ij}^k u_{ij}^k = l \cdot c_{ij} + (1 - l) \cdot \sum_{k\in K : u_{ij}^k \neq 0} p^k d_{ij}^k$, which is a convex combination of the two values c_{ij} and $\sum_{k\in K : u_{ij}^k \neq 0} p^k d_{ij}^k$. As the solution defined by variables y, u^k and x^k is optimal, we obtain $c_{ij} = \sum_{k\in K : u_{ij}^k \neq 0} p^k d_{ij}^k$. Hence, the extreme point solution $y_{ij} = 1$ and $u_{ij}^k = 0$ is also an optimal solution.

Model (2.1) is a two-stage stochastic optimization problem. Therefore, one can quantify the value of the proposed stochastic model (2.1) by the well-known “value of the stochastic solution” [4]. The idea of this textbook method is to compute first-stage cuts by replacing the second-stage uncertainties in model (2.1) by their expected values; the resulting deterministic optimization problem is called an expected value problem. The obtained first-stage cuts are then “evaluated” in the two-stage stochastic framework of model (2.1) and the resulting objective function value is compared with z^*. Note that any first-stage cut can be extended to a valid cutset, separating all K terminals from the single source s, because of the possibility to cut edges in the second stage.

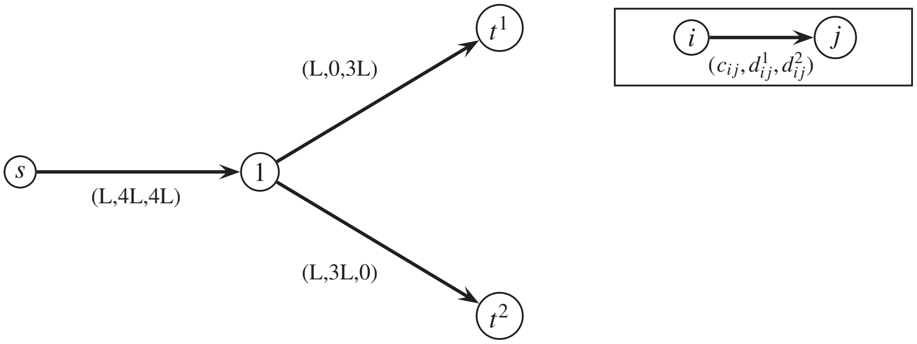

FIGURE 2 Instance with "value of the stochastic solution" of L

Since the stochastic model possesses possibly K different terminal nodes, the expected value problem separates s from all terminals (the technology matrix and the right-hand-side of model (2.1) are uncertain). Consequently, all second-stage cuts in a solution of the expected value problem cut all K second-stage edges. This may lead to an arbitrarily large "value of the stochastic solution". For such an example, consider Figure 2 with two scenarios of equal probability and some $L \in \mathbb{N}$. The optimal stochastic solution cuts edge $(1, t^1)$ for scenario 1 and edge $(1, t^2)$ for scenario 2, at 0 cost. The optimal solution of the expected value problem cuts edge $(s, 1)$ in the first stage at cost L . This leads to a value of the stochastic solution of L.

2.1 | Undirected graphs

Let $G(V, E)$ be an undirected graph. For undirected graphs, we call each $ij \in E$ an edge. In order to apply our results for directed graphs, we define the appropriate directed graph $G(V, A^{\mathrm{u}})$, associated with $G(V, E)$, with

$$A^{\mathrm{u}} := \{ij | ij \in E \text{ or } ji \in E\}$$

together with the corresponding arc cost c_{ij} and d_{ij}^k for all $ij \in A^{\mathrm{u}}$ and $k \in \{1, \ldots, K\}$.

The (deterministic) minimum $s-t$ cut problem formulation (1.2a)–(1.2d) for the corresponding directed graph $G(V, A^{\mathrm{u}})$ remains valid. However, the stochastic two-stage minimum $s - t^k$ cut problem formulation requires a slight adjustment: The first-stage cuts y_{ij} for $ij \in E$ need to affect both copies of edges $ij \in A^{\mathrm{u}}$ and $ji \in A^{\mathrm{u}}$; that is,

$$y_{ij} = y_{ji}, \qquad \forall ij \in E. \tag{2.3}$$

Equation (2.3) is not necessary for the deterministic minimum $s-t$ cut problems, as only one of the two arcs ij or ji is contained in any optimal cut (for $c_{ij} > 0$). In contrast, in the two-stage version, different directed arcs ij or ji might be cut for different scenarios k. Because an undirected graph does not distinguish between the two, we require additional constraints (2.3).

To improve computational performance, we make three adjustments:

1. Remove arcs is from A^{u}, for all $i \in V$.
2. Remove arcs $t^k i$ from A^{u}, for all $k = 1, \ldots, K$, $i \in V$ if $t^\kappa = t^{\kappa-1}$ for all $\kappa = 2, \ldots, K$, that is, all terminal nodes are identical.
3. Keep only one copy of the decision variables y_{ij} and y_{ji}; that is, replace inequality (2.1b) by

$$u_{ij}^k + y_{ij}\mathcal{I}_{i<j} + y_{ji}\mathcal{I}_{j<i} \geq x_j^k - x_i^k, \qquad \forall ij \in A^{\mathrm{u}}, \quad k = 1, \ldots, K, \tag{2.4}$$

where indicator function $\mathcal{I}_{a<b} = 1$ if $a < b$ and 0 otherwise.

The three improvements above can also be applied to directed graphs.

2.2 | General case: total unimodularity is lost

It is natural to consider whether variables x^k can be relaxed as well, similar to the deterministic case. This leads us to the discussion of whether the constraint matrix (2.1b)–(2.1d) is TU. In this section, we show via a counterexample that the constraint matrix of the two-stage stochastic minimum $s-t$ cut problem loses the total unimodularity property when extended from the deterministic case.

Recall that a matrix $\mathbb{A}$ is TU, if the determinant of each square sub-matrix of $\mathbb{A}$ has the value 0, 1, or -1. In order to discuss total unimodularity of the constraint matrix (2.1b)–(2.1d), it suffices to consider the matrix defined by (2.1b), due to the following lemma.

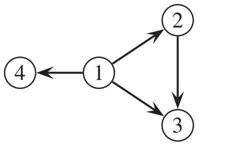

FIGURE 3 The matrix corresponding to this graph defined through constraints (2.1b)–(2.1d) is not TU for $K \geq 2$

Lemma 1. ([35]). *Let $\mathbb{A}$ be a matrix with $\{\pm 1, 0\}$ entries and $\mathbb{B}$ be the matrix where one row with exactly one entry with value $+1$ or -1 is added to $\mathbb{A}$. Then, matrix $\mathbb{B}$ is TU, if and only if $\mathbb{A}$ is TU.*

We now re-write the constraints in the form $\widetilde{x} \leq b$. To this end, let y be the vector of variables y_{ij}, u^k be the vector of variables u_{ij}^k, and x^k be the vector of variables x_i^k. Furthermore, let $\mathbb{B}$ be the arc-node incidence matrix of graph G with dimension $m \times n$, let $\mathbb{I}_m$ be the identity matrix with dimension $m \times m$ and $\mathbf{0}$ be the matrix with all entries as 0 (with an appropriate dimension). Finally, let $\widetilde{x} = (u^1, \dots, u^K, y, x^1, \dots, x^K)^{\top}$, where $(\cdot)^{\top}$ defines the transpose operator. This enables us to re-write inequality (2.1b) as

$$Ax^{\top} = \left(\begin{array}{cccc|c|cccc} -\mathbb{I}_m & \mathbf{0} & \cdots & \mathbf{0} & -\mathbb{I}_m & \mathbb{B} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & -\mathbb{I}_m & \vdots & \mathbf{0} & -\mathbb{I}_m & \mathbf{0} & \mathbb{B} & \vdots & \mathbf{0} \\ \vdots & \cdots & \ddots & \vdots & \vdots & \vdots & \cdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & -\mathbb{I}_m & -\mathbb{I}_m & \mathbf{0} & \mathbf{0} & \cdots & \mathbb{B} \end{array}\right) \left(\begin{array}{c} u^1 \\ u^2 \\ \vdots \\ u^K \\ \hline y \\ \hline x^1 \\ x^2 \\ \vdots \\ x^K \end{array}\right) \leq 0. \tag{2.5}$$

For two matrices $\mathbb{M}_1$ and $\mathbb{M}_2$ with the same number of rows, we denote by $\mathbb{M}_1 | \mathbb{M}_2$ the matrix obtained by "gluing" $\mathbb{M}_1$ to the left of $\mathbb{M}_2$. Then, we make use of the following lemma.

Lemma 2. ([37]). *Let $\mathbb{A}$ be an $n \times m$ matrix. $-\mathbb{I}_n | \mathbb{A}$ is TU, if and only if $\mathbb{A}$ is TU.*

With Lemma 2, it suffices to consider the following matrix

$$\mathbb{C} = \left(\begin{array}{c|cccc} -\mathbb{I}_m & \mathbb{B} & \mathbf{0} & \cdots & \mathbf{0} \\ -\mathbb{I}_m & \mathbf{0} & \mathbb{B} & \vdots & \mathbf{0} \\ \vdots & \vdots & \cdots & \ddots & \vdots \\ -\mathbb{I}_m & \mathbf{0} & \mathbf{0} & \cdots & \mathbb{B} \end{array}\right). \tag{2.6}$$

We now consider the graph with four nodes and four arcs shown in Figure 3. We note that the graph does not contain a directed cycle (but is also not a tree), and both source and sink nodes are not marked because this is irrelevant for the TU property.

For $K = 2$, the matrix $\mathbb{C}$ for Figure 3 is as follows.

$$\mathbb{C} = \begin{array}{c} \begin{array}{cccc|cccc|cccc} (1,4) & (1,2) & (1,3) & (2,3) & 1 & 2 & 3 & 4 & 1 & 2 & 3 & 4 \end{array} \\ \left(\begin{array}{cccc|cccc|cccc} -1 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ \hline -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \end{array}\right) \end{array}, \tag{2.7}$$

where the first four columns correspond to the arc variables y_{ij}, columns five to eight correspond to the node variables x_i^1 for the first scenario, and the last four columns to variables x_i^2 for the second scenario. Matrix $\mathbb{C}$ in (2.7) is not TU as the square

TABLE 1 Convex hull for different instances corresponding to Figure 3

Source node	Sink node scenario 1	Sink node scenario 2	# integral extreme points	# fractional extreme points
1	2	2	25 000	0
1	2	3	20 900	0
1	2	4	22 748	0
1	3	3	19 000	0
1	3	4	19 712	0
1	4	4	22 120	0

submatrix

$$\mathbb{E} = \begin{array}{c} \begin{array}{cccccc} (1,4) & (2,3) & 1 & 2 & 1 & 3 \end{array} \\ \left(\begin{array}{cc|cc|cc} -1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 0 & 0 \\ \hline -1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & -1 & 0 & 0 & 0 & 1 \end{array}\right) \end{array} \tag{2.8}$$

has determinant −2. Thus, in the general case the constraint matrix (2.1b)–(2.1d) is not TU.

Note that TU is just a sufficient condition for integral polyhedra. Through the example above, we see that the TU property of the deterministic $s - t$ cut problem is lost, in general. Despite the loss of the TU property, solutions of the LP-relaxation of model (2.1) might still be integral. This is important to note in light that we observe that many of the random instances yield integer solutions from their LP relaxations; see Section 6. In this context, consider Table 1. The table lists the number of extreme points of the corresponding polytope for different choices of source and sink nodes. The extreme points are calculated with the software PORTA [6]. As evident from the results, all extreme points are 0-1 valued. This shows that the polyhedra are integral! Therefore, any choice of arc costs leads to an integral optimal solution even though the constraint matrix is not TU.

2.3 | Trees: total unimodularity is preserved

In this section, we show that the constraint matrix A of (2.1b)–(2.1d) is TU if the underlying graph G is a tree; that is, the graph does not contain any (undirected) circuits. Specifically, we consider directed, connected graphs G, which are out-rooted trees. As such, G has exactly $n - 1$ arcs for n nodes.

As a by-product, we learn that the two-stage stochastic minimum $s - t$ cut problems for trees are single commodity flow problems on a transformed graph. Furthermore, in Section 4, we discuss a linear time algorithm for the two-stage stochastic minimum $s - t$ cut problem. All this implies that the two-stage stochastic minimum $s - t$ cut problem is polynomially solvable for trees—just as the demand robust minimum $s - t$ cut problem is [15].

We have already seen with Figure 3 that, in general, the constraint matrix (2.1b)–(2.1d) is not TU. The underlying reason is that the graph contains an undirected cycle. Forbidding undirected cycles for graphs leads to trees—and this property of a tree is exactly what we need in order to prove that the corresponding constraint matrix is TU.

Consider the following matrix

$$\mathbb{D} := \left(\begin{array}{cccc|c} \mathbb{B}^{\top} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbb{B}^{\top} & \vdots & \mathbf{0} & \mathbf{0} \\ \vdots & \cdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbb{B}^{\top} & \mathbf{0} \\ \hline \mathbb{I}_{n-1} & \mathbb{I}_{n-1} & \cdots & \mathbb{I}_{n-1} & \mathbb{I}_{n-1} \end{array}\right) \tag{2.9}$$

with node-arc incidence matrix $\mathbb{B}^{\top}$ of graph G with dimension $n \times (n-1)$ and identity matrix $\mathbb{I}_{n-1}$ with dimension $(n-1) \times (n-1)$. Matrix $\mathbb{D}$ has been obtained from $\mathbb{C}$ by the following operations. First, matrix $\mathbb{C}$ has been transposed. In the transposed graph, second, the last $n - 1$ rows are multiplied by -1 and third, the last $n - 1$ columns are added (containing exactly one $+1$ entry). All three operations do not alter the TU property (see Lemma 9.2.2 in [35]). In fact, matrix $\mathbb{D}$ is TU if and only if $\mathbb{C}$ is TU. Therefore, $\mathbb{D}$ is TU if and only if A is TU.

Matrix $\mathbb{D}$ is the constraint matrix of a multicommodity flow problem on the graph corresponding to node-arc incidence matrix $\mathbb{B}^\top$; see for example [32, 34]. Because G is a tree with one root node, the multicommodity flow problem has one supply node, s, and K demand nodes, t^k, $k = 1, \ldots, K$. In particular, G does not contain any circuits.

Soun and Truemper [32] describe a general procedure for transforming a multicommodity flow problem to a single commodity flow problem which preserves TU. Because such a transformation is not possible in general (consider our counterexample in Section 2.2), the first step of the procedure checks whether the graph is transformable. This check iteratively removes nodes from 2-connected graphs until a cycle with two arcs is left (see "Step 1 (Analysis)" in the transformation algorithm in [32]). Because tree G is not 2-connected, G has to be decomposed into its 2-connected components, which are just single nodes (this is where we exploit the assumption that G is a tree). Each of the single nodes is trivially transformable (there is no flow to send) and with that the multicommodity flow problem (see page 358 in [32]). Therefore, the multicommodity flow problem is equivalent to some single commodity flow problem, whose constraint matrix is TU. By transformability, matrix $\mathbb{D}$ is TU [32]. This completes our proof and shows that matrix A of (2.1b)–(2.1d) is TU for trees.

3 | COMPUTATIONAL COMPLEXITY

The fact that the constraint matrix of the two-stage stochastic minimum $s-t$ cut problem loses its total unimodularity (see Section 2.2) raises the question about the theoretical computational complexity of the problem. In this section, we show that the stochastic programming extension of the polynomially solvable (deterministic) minimum $s-t$ cut problem becomes $\mathcal{NP}$-hard in general. This is consistent with similar observations for the two-stage stochastic extensions of the minimum spanning tree [13] and the maximum weight matching [23] problems. In this section for simplicity of further discussion, we consider the undirected version of the problem. We define the decision version of two-stage stochastic $s-t$ cut problem as follows.

Definition 2. (Decision version).

Instance: An undirected graph $G = (V, E)$ with node set V, edge set E, root node $s \in V$, K scenarios and bound $C < \infty$. The kth scenario consists of a single terminal t^k and has probability p^k, where $0 < p^k < 1$, of being realized. Edge $ij \in E$ has nonnegative cost c_{ij} in the first stage and nonnegative cost d_{ij}^k in the recourse stage (or second stage) if the kth scenario is realized.

Question: Is there a set of arcs E_0 to be cut in the first stage and for each scenario k, an edge set E_k to be cut in the recourse stage, if scenario k is realized, such that removing $E_0 \cup E_k$ from the graph G disconnects s from the terminal t^k, while the expected cost of cutting $c := \sum_{ij \in E_0} c_{ij} + \sum_{k=1}^{K} p^k \sum_{ij \in E_k} d_{ij}^k$ over all scenarios does not exceed C, that is, $c \le C$?

We call the edge set $E_0 \cup E_k$ a *feasible cut* for scenario k, if the removal of $E_0 \cup E_k$ from the graph G disconnects the node s from the terminal node t^k. In our reduction, we use the multiterminal cut (MC) problem.

Definition 3. (Multiterminal cut, [8]).

Instance: An undirected graph $G(V, E)$ with node set V, edge set E, a set $S = \{s_1, s_2, \ldots, s_k\} \subseteq V$ of k specified terminals, a positive weight w_{ij} for each edge $ij \in E$ and a bound $B < \infty$.

Question: Is there a subset of arcs $\overline{E} \subseteq E$ with $\sum_{e \in \overline{E}} w_{ij} \le B$ such that the removal of $\overline{E}$ from E disconnects each terminal from all others?

An edge set $\overline{E} \subseteq E$ is called a *feasible cut* for MC, if the removal of $\overline{E}$ from E disconnects each terminal from all others. We need the following complexity result.

Lemma 3. ([8]). *The multiterminal cut problem for* $k = 3$ *and arbitrary graphs is* $\mathcal{NP}$*-complete even if all edge weights are equal to* 1.

Note that for $k = 2$ the MC problem reduces to the standard minimum $s-t$ cut problem, which can be solved in polynomial time.

We now establish the strongly $\mathcal{NP}$-completeness of the two-stage stochastic $s-t$ cut problem by reducing the MC problem with $k = 3$ to it. We are given an instance of the MC problem with $G = (V, E)$, respective weights w_{ij} for each $ij \in E$, $S = \{s_1, s_2, s_3\}$ and bound B. Without loss of generality (i.e., the MC problem remains NP-complete), we assume that arcs $s_1 s_2$ and $s_1 s_3$ do not exist in G; that is, $s_1 s_2, s_1 s_3 \notin E$.

Next we construct an instance of the two-stage stochastic $s-t$ cut problem such that there is a one-to-one correspondence between their respective solutions. We define graph $G_{MC} = \widetilde{G}(\widetilde{V}, \widetilde{E})$ as follows:

- Let $\widetilde{V} = V \cup \{t\}$ and $\widetilde{E} = E \cup \{s_1s_2, s_1s_3, s_2t, s_3t\}$. In other words, we add an extra node and four additional arcs into the original graph for the MC problem.
- Let the first-stage edge capacities be $c_{ij} = w_{ij} \, \forall \;\, ij \in E$ and $c_{ij} = +\infty$ for $ij \in \{s_1s_2,\, s_1s_3,\, s_2t,\, s_3t\}$.
- Let the second-stage edge capacities be $d^1_{ij} = d^2_{ij} = +\infty \forall \; ij \in E$, $d^1_{s_1s_2} = d^1_{s_3t} = d^2_{s_1s_3} = d^2_{s_2t} = +\infty$, and $d^1_{s_1s_3} = d^1_{s_2t} = d^2_{s_1s_2} = d^2_{s_3t} = 0$.
- Let the root node be $s := s_1$ and the terminal node be t for both scenarios.
- Let the probability of each scenario be given by $p_1 = p_2 = 1/2$.

An example and the corresponding transformation is shown in Figure 4. The MC instance in Figure 4A is a "YES" instance for $B \geq 9$, with the feasible cutset $\overline{E} = \{s_15, s_14, s_24, s_2s_3\}$. This set is marked by the gray lines. Figure 4B shows the transformed graph G_{MC} with a cut having weight 9.

Lemma 4. *An instance for the multiterminal mut problem with bound B is a "YES" instance, if and only if the transformed graph G_{MC} is a "YES" instance for the two-stage stochastic $s - t$ cut problem with cost bound $C = B$.*

Proof. "⇒" Let the MC problem have the feasible cutset $\overline{E}$ with $\sum_{ij \in \overline{E}} w_{ij} \leq B$. The first-stage cutset for G_{MC} is $\overline{E}$ as well, since for the first scenario, arcs s_1s_3 and s_2t are cuts in the second stage and for the second scenario arcs s_1s_2 and s_3t are a cut in the second stage. In this way, the cut weight is $c \equiv w$. This leads to a feasible cut for G_{MC} for the two-stage $s - t$ cut problem for both scenarios with weight $c \leq C = B$.

"⇐" Suppose the solution of the constructed two-stage stochastic minimum $s - t$ cut problem is given by the edge set $\widetilde{E}_0$ for the first stage and edge sets $\widetilde{E}_1$ and $\widetilde{E}_2$ for scenarios one and two in the second stage, respectively, and cut weight C. Because $C < \infty$, any feasible two-stage $s - t$ cut has a finite total capacity. We then have the following facts:

- $\widetilde{E}_0 \subseteq E$ since $c_{ij} = +\infty$ for any $ij \in \widetilde{E} \setminus E$.
- Any $ij \in E$ cannot belong to either $\widetilde{E}_1$ or $\widetilde{E}_2$.
- We may assume that $\widetilde{E}_1 = \{s_1s_3,\, s_2t\}$ and $\widetilde{E}_2 = \{s_1s_2,\, s_3t\}$ since the respective capacities are zero, while all other capacities are $+\infty$.
- In scenario one, since $d^1_{s_1s_2} = d^1_{s_3t} = +\infty$, arcs s_1s_2 and s_3t are not cut. Hence, $\widetilde{E}_0$ must contain arcs that completely disconnect s_1 from s_3; because s_1 is connected to s_2 in scenario one, s_2 must be disconnected from s_3 as well.
- In scenario two, since $d^2_{s_1s_3} = d^2_{s_2t} = +\infty$, arcs s_1s_3 and s_2t are not cut. Hence, $\widetilde{E}_0$ must contain arcs that completely disconnect s_1 from s_2; because s_1 is connected to s_3 in scenario two, s_2 must be disconnected from s_3 as well.

Therefore, any two-stage stochastic $s - t$ cut with a finite total capacity must contain $\widetilde{E}_0$ that completely disconnects s_1, s_2 and s_3 from each other; that is, $\widetilde{E}_0$ is a multiterminal cut in the original graph $G(V, E)$. Moreover, since the capacities of arcs in $\widetilde{E}_1$ and $\widetilde{E}_2$ are zero, minimizing the total capacity of the two-stage $s - t$ cut corresponds to minimizing the weight of the multiterminal cut. Thus, $\widetilde{E}_0$ is a feasible cut for MC with weight $B = C$. ■

This allows us to prove our main result.

Theorem 1. *The decision version of the two-stage stochastic $s - t$ cut problem is $\mathcal{NP}$-complete in the strong sense even for two scenarios and the same terminal node for both scenarios.*

Proof. The two-stage stochastic $s - t$ cut problem is in $\mathcal{NP}$, as a nondeterministic algorithm needs only to guess which arcs to cut, check if this leads to a feasible cut for each scenario, and check if the cost of the cut is less than or equal to C.

The given transformation from MC to the two-stage stochastic $s - t$ cut problem via graph G_{MC} is valid according to Lemma 4. By replacing the weights $+\infty$ by $B + 1$, the node set, the edge set, and the edge weights of the constructed graph G_{MC} are linearly bounded in the input size of MC. Thus, the transformation can be done in (strongly) polynomial time. ■

Corollary 1. *The directed version of the two-stage stochastic $s - t$ cut problem is also $\mathcal{NP}$-complete in the strong sense. To see this, one can use the same reduction as for the undirected case by preserving the direction of the arcs to be cut in the construction presented in the proof of Lemma* 4.

4 | LINEAR RUNNING TIME ALGORITHM FOR TREES

As we discussed in Section 2.3, if graph G is a tree, then the constraint matrix (2.1b)–(2.1d) is TU. This fact indicates that the two-stage stochastic minimum $s - t$ cut problem is polynomially solvable if G is tree. Furthermore, we show next that the problem admits a linear time solution algorithm.

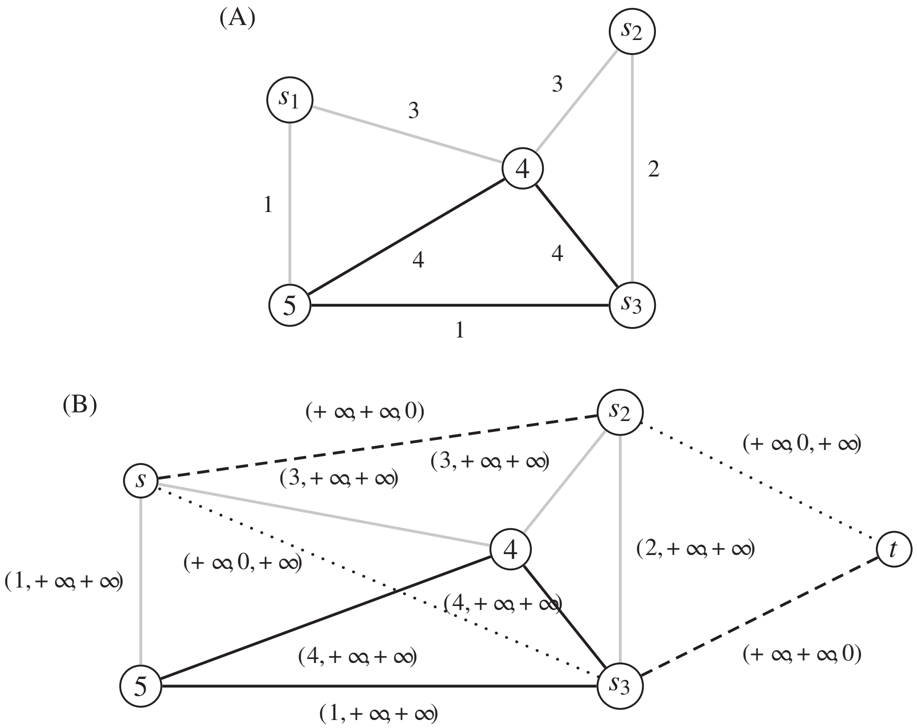

FIGURE 4 MC instance and corresponding instance G_{MC} for the two-stage stochastic minimum $s-t$ cut problem. The legend for (B) is the same as in Figure 1, but rather for undirected arcs. (A) "YES" instance for MC problem for $B \geq 9$ with cut; gray arcs are cut. (B) Corresponding instance for the two-stage stochastic minimum $s-t$ cut problem with cut

Consider now the following transformation. Given an instance of the two-stage stochastic minimum $s-t$ cut problem on directed graph $G = (V, A)$ with the notation of Definition 1, construct a directed graph $\overline{G} = (\overline{V}, \overline{A})$ as follows:

- Add one additional node $\overline{T}^k$ to V for each scenario k; that is, $\overline{V} := V \bigcup_{k=1}^{K} \{\overline{T}^k\}$.
- Add one arc between terminal t^k and $\overline{T}^k$ to A for each scenario k; that is, $\overline{A} = A \bigcup_{k=1}^{K} \{t^k\overline{T}^k\}$.
- Define weights for $ij \in A$ as the first-stage costs; that is, $w_{ij} = c_{ij}, \forall\, ij \in A$.
- Define weights for $t^k\overline{T}^k$ as follows. Let P^k be the (unique) path from the source s to t^k, and e^k be one least cost arc in scenario k (along this path P^k). Then $w_{t^k\overline{T}^k} = p^k d^k_{e^k}, \forall k = 1, \dots, K$.

Graph $\overline{G}$ remains a tree by construction. Now, finding a (deterministic) minimum cut in $\overline{G}$, which separates s from all terminals $\overline{T}^k$, can be done via a linear time dynamic programming algorithm (use a backward recursion from the terminals to s and update the arcs to be cut). Any such cut in $\overline{G}$ corresponds, then, to a cut in G (and vice-versa) with the same cost as follows: if arc $ij \in A$ for $\overline{G}$ is a cut, then in the first stage, ij is a cut in G; if arc $ij = t^k\overline{T}^k \in \overline{A} \setminus A$ is a cut, then in the k-scenario, the corresponding arc e is a cut. Hence, we have the following lemma:

Lemma 5. *If graph G is a tree, then the two-stage stochastic minimum $s-t$ cut problem can be solved in linear time.*

Figure 5 shows the transformation for an instance with three scenarios, having equal probability. First, the three nodes $\overline{T}^1$, $\overline{T}^2$, and $\overline{T}^3$ and the three arcs $t^1\overline{T}^1$, $t^2\overline{T}^2$, and $t^3\overline{T}^3$ are added for the three scenarios. The arcs cost for all arcs ($s2$, $2t^1$, $2t^2$, $s3$, $3t^3$) of the original graph are given by the first-stage arc cost c_{ij}. The weights for the additional arcs $t^1\overline{T}^1$, $t^2\overline{T}^2$, and $t^3\overline{T}^3$ are given by $\frac{1}{3}\min\{7, 6\} = 2$, $\frac{1}{3}\min\{6, 7\} = 2$ and $\frac{1}{3}\min\{3, 4\} = 1$, respectively. The optimal cuts are marked accordingly.

Adjusting the formulation of the deterministic minimum $s-t$ cut problem (1.2a)–(1.2d) to the cut problem for graph $\overline{G}$, we obtain:

$$\min \sum_{ij \in \overline{A}} w_{ij} y_{ij} \tag{4.1a}$$

$$\text{s.t.} \quad y_{ij} \geq x_j - x_i, \qquad \forall ij \in \overline{A}; \tag{4.1b}$$

$$x_s = 0; x_{t^k} = 1, \qquad \forall k = 1, \dots, K; \tag{4.1c}$$

$$x_i, y_{ij} \in [0, 1], \qquad \forall i \in \overline{V}, \quad ij \in \overline{A}. \tag{4.1d}$$

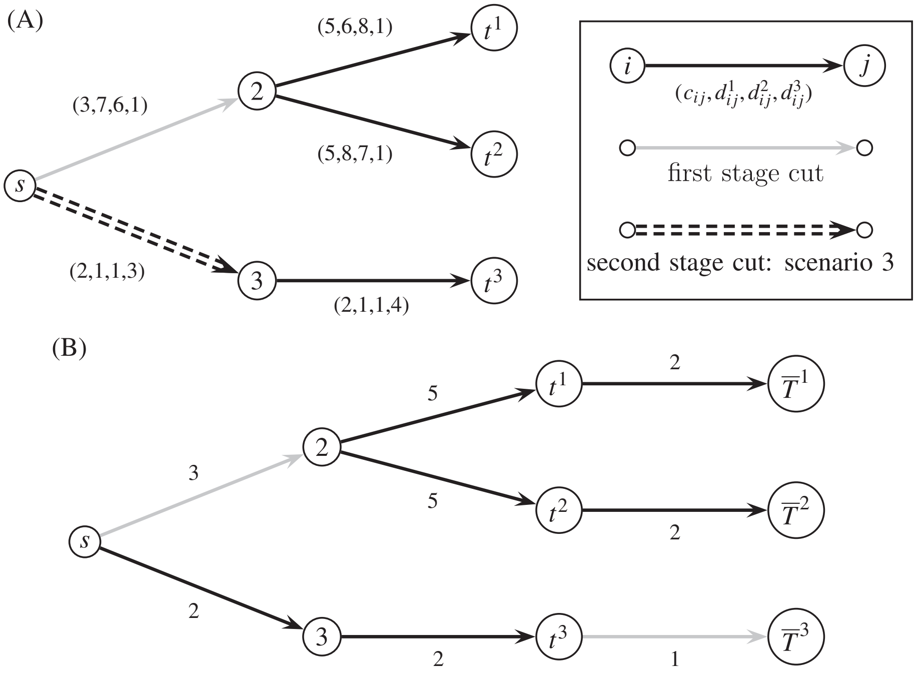

FIGURE 5 Two-stage stochastic min-cut instance for a tree and corresponding deterministic min-cut instance. (A) Stochastic min-cut instance for tree with three scenarios. (B) Transformed tree instance of deterministic min-cut problem

The constraint matrix defined by (4.1b)–(4.1c) is TU. However, this does not (at least, in an obvious manner) imply that the constraint matrix is TU for the tree G as well.

Finally, we should note that the approach discussed in this section is similar in spirit to the one in [14] for the two-stage robust min-cut problem, as the latter also admits a polynomial time solution whenever G is a tree.

5 | BENDERS DECOMPOSITION

The y variables in inequality (2.1b) represent the $s - t^k$ cuts in the first stage. In other words, the y variables connect the K stages. Thus, if variables y are fixed to 0 or 1, then problem (2.1a)–(2.1e) decomposes into K separate minimum $s - t$ cut problems of the type (1.2a)–(1.2d). Each of the K optimization problems is a linear program, because of its total unimodularity property. This structure suggests a Benders decomposition approach [3], where the master problem contains the y variables and the K subproblems are minimum $s - t$ cut problems for trial values $\overline{y}$ obtained from the master problem. By using such a proposed decomposition, the underlying TU structure of each of the K scenarios is revealed and can be exploited.

Benders decomposition is also known as the L-shaped method in the stochastic programming community [31]. We assume that the reader is familiar with Benders decomposition, otherwise we refer to [26, 27] and the references therein. Note that our Benders subproblems are LPs because of the TU property of the $s - t^k$ min-cut problems. Thus, we do not require methods from two-stage stochastic integer programming, which are very hard to solve problems when second-stage integrality requirements are present.

We start with the description of the sub-problem of the Benders decomposition algorithm. For a given trial solution $\overline{y}$, obtained from the master problem, we have K subproblems, one for each scenario k, as follows

$$z_k^*(\overline{y}) = \min \quad \sum_{ij \in A} d_{ij}^k u_{ij}^k \tag{5.1a}$$

$$\text{s.t.} \quad u_{ij}^k - x_j^k + x_i^k \geq -\overline{y}_{ij}, \qquad \forall ij \in A; \tag{5.1b}$$

$$x_s^k = 0; \tag{5.1c}$$

$$x_{t^k}^k = 1; \tag{5.1d}$$

$$x_i^k, u_{ij}^k \in [0,1], \qquad \forall i \in V, \quad \forall ij \in A. \tag{5.1e}$$

We note that all K subproblems (5.1a)–(5.1e) are feasible for any (binary) $\overline{y}$, which eliminates the need for Benders feasibility cuts. We add K optimality cuts, one for each sub-problem, (the so-called "multicut version" of Benders decomposition). For $k = 1, \ldots, K$, the cut is derived as

$$\eta^k \geq \sum_{ij \in A} (-\pi_{ij}^k) y_{ij} + \varpi^k; \tag{5.2a}$$

$$\varpi^k = z_k^*(\overline{y}) + \sum_{ij \in A} \pi_{ij}^k \overline{y}_{ij}, \tag{5.2b}$$

where π_{ij}^k is an optimal dual solution associated with constraint (5.1b) and ϖ^k is the cut constant.

To improve computational performance, when solving the Benders subproblems (5.1), we apply Proposition 3; that is, constraints (5.1b) are eliminated whenever trial solution $\overline{y}_{ij} = 1$.

The master problem is then derived as

$$\underline{z} = \min \quad \sum_{ij \in A} c_{ij} y_{ij} + \sum_{k=1}^{K} p^k \eta^k \tag{5.3a}$$

$$\text{s.t.} \quad \eta^k \geq \sum_{ij \in A} (-\pi_{ij}^k) y_{ij} + \varpi_l^k, \qquad \forall k = 1, \ldots, K, \quad \forall l = 1, \ldots, L; \tag{5.3b}$$

$$y_{ij} \in \{0, 1\}, \qquad \forall ij \in A, \tag{5.3c}$$

$$\eta \geq 0, \tag{5.3d}$$

where l is the index of the Benders iterations. Note that $\eta \geq 0$ because $\eta \geq z_k^*(\cdot)$ and $z_k^*(\overline{y}) \geq 0$ for any binary $\overline{y}$ due to the nonnegative second-stage arc costs d_{ij}^k.

The master problems yield a lower bound $\overline{z}$ on the optimal solution value z^*. A feasible solution $\overline{y}$ of the master problem provides an upper bound as:

$$\overline{z}(\overline{y}) = \sum_{ij \in A} c_{ij} \overline{y}_{ij} + \sum_{k=1}^{K} p^k z_k^*(\overline{y}). \tag{5.4}$$

We stop the Benders decomposition algorithm if the relative gap between the upper and the lower bound is less than a predefined tolerance $\varepsilon > 0$.

The resulting Benders algorithm is summarized in Algorithm 1. We start with the initialization in Step 1. We initialize the Benders iteration count, l, and set the upper bound $\overline{z}$ to infinity. If a feasible solution is known, then the upper bound takes on its objective function value. Next, we solve the master problem in Step 2. Its optimal decision variables are t he trial solution, $\overline{y}$, required to set up the Benders subproblem in the next step. The subproblem in Step 3 is tackled by solving K independent LPs. Together with the objective function value of the master problem, we compute and update the upper bound in Step 4. Because the master problem's optimal objective function value, $\overline{z}$, is a lower bound on z^*, we can check convergence of the algorithm in Step 5 by comparing lower and upper bounds. If convergence is achieved, then the algorithm terminates with an optimal solution. Otherwise, we compute the K Benders optimality cuts in Step 6. The algorithm starts over by solving the master problem in Step 2 again (with the K additional cuts), after the iteration counter is incremented in Step 7.

Algorithm 1. Benders decomposition for two-stage stochastic minimum $s-t$ cut problem (2.1).

```
Input: Graph $G(V, A)$, tolerance $\varepsilon > 0$
Output: Optimality cut, lower and upper bound on $z^*$
  1: $l = 0, \overline{z} = +\infty$.
  2: Solve master problem (5.3); obtain trial solution $\overline{y}$ and lower bound $\overline{z}$.
  3: Forall $k = 1, \ldots, K$: Solve sub-problem (5.1) with trial solution $\overline{y}$; obtain optimal objective function value $z_k^*(\overline{y})$, optimal
     dual solution $\pi^k$ associated with constraint (5.1b).
  4: Compute upper bound, $\overline{z} \leftarrow \min \left\{ \overline{z}, \sum_{ij \in A} c_{ij} \overline{y}_{ij} + \sum_{k=1}^{K} p^k z_k^*(\overline{y}) \right\}$
  5: If $\frac{\overline{z}-z}{\overline{z}} \leq \varepsilon$, then STOP. Return optimal min $s-t$ cut (first-stage cut, $\overline{y}$, and second-stage cuts, $u^k$), lower bound $\overline{z}$ and upper
     bound $\overline{z}$.
  6: For all $k = 1, \ldots, K$: Compute Benders optimality cut (5.2) using $z_k^*(\overline{y})$, $\pi^k$ and $\overline{y}$ and add the cut to the master problem
  7: $l \leftarrow l + 1$, go to Step 2.
```

We note from Algorithm 1 that we need to solve the Benders subproblem (5.1) in order to (i) compute an upper bound, and (ii) to compute a Benders optimality cut. For both tasks, we require the optimal objective function value, $z_k^*(\bar{y})$, and the duals, π^k, associated with constraint (5.1b). Algorithm 1 obtains these by solving K LPs in each iteration. Next, we discuss a way to more efficiently obtain $z_k^*(\bar{y})$ and π^k, via a direct calculation method. Such a direct cut calculation, if the problem admits it, can yield very efficient solution algorithms [24].

By the max-flow min-cut theorem, we know that the dual of the Benders subproblem (5.1) is a max-flow problem. The dual of problem (5.1) for a fixed scenario k and trial solution $\bar{y}$ of the master problem is as follows:

$$z_k^*(\bar{y}) = \max -\sum_{ij\in A} \bar{y}_{ij}\pi_{ij}^k + \bar{\pi}_{t^k}^k + \sum_{ij\in A} \lambda_{ij}^k + \sum_{i\in V} \theta_i^k \tag{5.5a}$$

$$\text{s.t.} \quad \pi_{ij}^k + \lambda_{ij}^k \leq d_{ij}^k, \qquad \forall ij \in A; \tag{5.5b}$$

$$\sum_{j:ij\in A} \pi_{ij}^k - \sum_{j:ji\in A} \pi_{ji}^k + \theta_i^k \leq 0, \qquad \forall i \in V \setminus \{s, t^k\}; \tag{5.5c}$$

$$\sum_{j:sj\in A} \pi_{sj}^k - \sum_{j:js\in A} \pi_{js}^k + \bar{\pi}_s^k + \theta_s^k \leq 0; \tag{5.5d}$$

$$\sum_{j:t^kj\in A} \pi_{t^kj}^k - \sum_{j:jt^k\in A} \pi_{jt^k}^k + \bar{\pi}_{t^k}^k + \theta_{t^k}^k \leq 0; \tag{5.5e}$$

$$\pi_{ij}^k \geq 0, \lambda_{ij}^k \leq 0, \qquad \forall ij \in A; \tag{5.5f}$$

$$\theta_i^k \leq 0, \qquad \forall i \in V; \tag{5.5g}$$

$$\bar{\pi}_s^k, \bar{\pi}_{t^k}^k \text{ free}, \tag{5.5h}$$

with dual variables π_{ij}^k, $\bar{\pi}_s^k$ and $\bar{\pi}_{t^k}^k$ associated with constraints (5.1b)-(5.1d) respectively, and dual variables λ_{ij}^k and θ_i^k associated with ≤ 1 constraints (5.1e) for variables x_i^k and u_{ij}^k, respectively. Note that, because of strong duality, the optimal objective function values of model (5.1) and model (5.5) are the same.

Recognizing its special structure, model (5.5) can be reformulated in the more typical max-flow form:

$$z_k^*(\bar{y}) = \max \sum_{j:jt^k\in \bar{A}} \pi_{jt^k}^k \tag{5.6a}$$

$$\text{s.t.} \quad \sum_{j:ij\in \bar{A}} \pi_{ij}^k - \sum_{j:ji\in \bar{A}} \pi_{ji}^k = 0, \qquad \forall i \in V \setminus \{s, t^k\}; \tag{5.6b}$$

$$\pi_{ij}^k \leq d_{ij}^k, \qquad \forall ij \in \bar{A}; \tag{5.6c}$$

$$\pi_{ij}^k \geq 0, \qquad \forall ij \in \bar{A}, \tag{5.6d}$$

with $\bar{A} := \{ij \in A | \bar{y}_{ij} = 0\}$.

As such, model (5.6) is an $s - t^k$ max-flow problem on graph $\hat{G} = (V, \bar{A})$ with *capacities* given by the kth scenario's cost. We remove arcs from the original graph, if they are included in the first-stage cut; that is, for $ij \in A$ with $\bar{y}_{ij} = 1$. This follows from Proposition 3.

Algorithm 1 mainly needs an update in Step 3. The subproblems are no longer solved as general LPs by a linear programming solver. Instead, the dual variables and the objective function value (both required for the Benders optimality cut in Step 6) are obtained by solving K max-flow problems, recognizing strong duality. Because the max-flow problems are the dual of the (primal) subproblems, we no longer have values for the second-stage cut variables u^k. Thus, if the algorithm terminates in Step 5, the optimal second-stage cut variables u^k need to be computed (e.g., by solving min-cut problem (5.1)), if desired.

We want to highlight the following observation. Because any (extreme point) solution of the max-flow problem (5.6) is integral (for integral second-stage arc costs d_{ij}^k), the cut coefficients and cut constant of the Benders optimality cut (5.2) are all integral as well. This, in turn, means that the optimal η^k in the Benders master problems are also integral. If the first-stage arc costs c_{ij} are integral, then the optimal objective function value z must change by 0 or $\min_{ij\in A, k=1,\ldots,K}\{c_{ij}, p^k \mid c_{ij} \neq 0\}$ between iterations. In other words, the improvement is either 0 or bound from below by the smallest nonzero objective function coefficient of the master problem. Note also that the number of iterations with no improvement is bounded by the number of multiple (binary) optimal solutions of the master problem. This is very atypical. This property of Algorithm 1 is significant because it implies that convergence is not slow when the lower bound gets close to the upper bound. This stands in contrast to most Benders decomposition algorithms where very slow convergence is observed when closing the optimality gap. Loosely speaking, in Benders decomposition, one typically observes improvements which converge to 0 with the number of iterations, while this cannot happen for Algorithm 1 provided arc costs c_{ij} and d_{ij}^k are integral.

6 | COMPUTATIONAL EXPERIMENTS

In this section, we benchmark the developed Benders decomposition algorithm for the two-stage stochastic minimum $s-t$ cut problem against the extensive formulation (2.1). The two algorithms are presented in Section 5 as implementation variants of Algorithm 1. "Benders+VLP" is the Benders algorithm where the subproblems are solved as vanilla LPs, while "Benders+max-flow" solves the subproblems via Ford-Fulkerson's max-flow algorithm. All resulting LP and MILP problems are solved with CPLEX 12.8.0 using the default settings. We use the C++ programming language and a standard Ford-Fulkerson implementation to solve the max-flow problems. For our parallel runs, we utilize the internal CPLEX parallel implementation when solving MILPs; the Benders algorithm implementations rely on the C++ threat class to solve the LPs and max-flow instances in parallel (on eight cores). We set a time limit of 10 000 seconds and a relative optimality gap of 0.001%. We use the LP relaxation as a lower bound and a sub-optimal solution as an upper bound for the extensive formulation and all Benders algorithm implementations. We exclude the computational times to obtain the lower and upper bounds when we report run times of the extensive formulation or the Benders algorithm implementations. All computations are performed on a 64bit Intel(R) Xeon(R) with 3.20 GHz and 16 GB RAM featuring eight cores.

It turns out that most arbitrary graphs for the two-stage stochastic minimum $s-t$ cut problem yield instances with a zero duality gap; that is, the LP relaxation yields integral solutions. To this end, we constructed thousands of random graphs by varying the following parameters: number of nodes, density of graph, number of scenarios, placement of terminal nodes and cost. We use a uniform random number generator to generate these parameters. In all of these instances, the LP relaxation yielded integral solutions. Thus, motivated by the transformation described in Figure 4 and the proof of Lemma 4, we construct two problem classes where optimal solutions are intuitive, yet the models are computationally challenging.

Note that both instance classes have some limitations regarding the uncertainty characterization. Instance class 1 has a limited random structure, since there is no uncertainty in the terminal node and the random arc cost parameters take only a limited number of values: 0, 1, $+\infty$. Instance class 2 is limited in terms of capturing uncertainty, because we can only solve instances with a very small number of scenarios (up to 30 scenarios). However, it is essential to consider a larger set of scenarios to have a better characterization of the underlying uncertainty.

6.1 | Instance class 1

Our first instance class is shown in Figure 6. It consists of the root s, a single terminal t, nodes $i_1, \ldots, i_N$ and nodes $j_1, \ldots, j_K$. K defines the number of (second-stage) scenarios. Each of the nodes $i_1, \ldots, i_N$ is connected to each node $j_1, \ldots, j_K$, and adjacent i nodes are connected to each other as well. The solid lines have a cost of 1 in the first stage and ∞ in the second stage, for all scenarios. The dashed lines have first-stage cost ∞. All dashed lines have a cost of 0 in the second stage for all scenarios, except for the following edges that have a cost of ∞ for scenario $k=1, \ldots, K$: between nodes s and j_m, $m \neq k$, and between nodes j_k and t. Other edge costs are possible as well, but we choose this cost structure to facilitate an analytical expression for the optimal total cost. All scenarios are equally likely. For our computations, we set 100 000 instead of the infinite edge cost (smaller objective function coefficients are possible, but tend to yield worse performance for the extensive formulation). An optimal first-stage solution cuts edges between the i and the s nodes yielding an optimal value of $N \cdot K$. The optimal second-stage solution in scenario k cuts the following edges: the single edge between nodes s and j_k, and the $K-1$ edges between nodes j_m, $m \neq k$ and t. The optimal cost for this instance is $N \cdot K$.

Consequently, the first instance class shares the same terminal node for all scenarios k; the second-stage edge cost differs significantly between 0 and infinity. The graph size changes with a different number of scenarios K. Parameters N and K completely determine the instance. Therefore, we have only one instance per parameter combination.

The computational results for our first instance class are summarized in Tables 2 and 3. Consider now Table 2. The first four columns characterize the instances. RMILP is the optimal objective function value of the LP relaxation and MILP is the optimal objective function value z^*. The next six columns provide the computational times (in seconds) of the three methods tested: the extensive formulation, Benders+VLP, Benders+max-flow. Each of the three algorithms is executed sequentially and in parallel (eight cores). The next two columns report the lower bound for instances which could not be solved to proven optimality within the time limit. The next four columns with label "# Iterations" state the number of iterations of the four Benders algorithms. Finally, the last column shows the speedup of the Benders+max-flow algorithms over the Benders+VLP algorithms

TABLE 2 Computational results for instance class 1

				Time (s)						Lower bound		# Iterations					
Instance				Sequential	Parallel	Sequential		Parallel		Sequential	Parallel	Sequential		Parallel		Speedup	
K	*N*	RMILP	MILP	Extensive form.		Benders+ VLP	Benders+ max-flow	Benders+ VLP	Benders+ max-flow	Extensive form.		Benders+ VLP	Benders+ max-flow	Benders+ VLP	Benders+ max-flow	Sequential	Parallel
5	10	30	50	**0.1**	0.2	0.6	**0.1**	0.2	**0.1**	=	=	25	31	25	31	6	2
	20	60	100	**0.3**	**0.3**	2.5	0.3	0.8	**0.2**	=	=	55	61	55	61	8	4
	30	90	150	**0.6**	0.7	5.5	0.6	1.7	**0.5**	=	=	75	91	75	91	9	3
	40	120	200	1.4	**0.7**	5.4	0.6	1.9	**0.5**	=	=	92	121	92	121	9	3
	50	150	250	1.6	**1.5**	10.8	1.0	3.4	**0.8**	=	=	113	151	113	151	10	4
	60	180	300	2.1	**1.8**	19.6	1.7	6.1	**1.4**	=	=	146	181	146	181	11	4
	70	210	350	2.5	**1.8**	38.9	2.6	11.9	**2.1**	=	=	166	211	166	211	15	5
	80	240	400	3.3	**3.2**	56.8	3.7	19.3	**3.1**	=	=	185	241	185	241	15	6
	90	270	450	3.8	**3.7**	87.6	5.1	30.0	**4.4**	=	=	205	271	205	271	17	6
	100	300	500	5.8	**5.4**	123.0	6.9	43.6	**5.8**	=	=	230	301	230	301	17	7
10	10	55	100	**0.3**	0.4	1.6	**0.1**	0.4	**0.1**	=	=	28	31	28	31	16	4
	20	110	200	**0.8**	0.9	4.7	0.4	1.2	**0.3**	=	=	44	61	44	61	11	4
	30	165	300	**1.9**	2.3	10.9	0.8	2.8	**0.7**	=	=	63	91	63	91	13	4
	40	220	400	**20.1**	24.4	24.6	1.6	5.5	**1.2**	=	=	91	121	91	121	15	4
	50	275	500	**30.2**	36.4	43.0	2.9	10.5	**2.1**	=	=	106	151	106	151	14	5
	60	330	600	**43.0**	62.9	71.6	4.2	16.9	**3.4**	=	=	127	181	127	181	17	5
	70	385	700	**62.4**	102.7	143.6	10.1	41.3	**8.6**	=	=	147	211	147	211	14	4
	80	440	800	**69.2**	179.7	295,6	13.8	64.5	**11.8**	=	=	167	241	167	241	21	5
	90	495	900	**84.8**	261.2	482.9	18.0	108.5	**15.3**	=	=	211	271	211	271	26	7
	100	550	1000	**139.8**	413.0	663.4	24.9	142.2	**21.3**	=	=	222	301	222	301	26	6
15	10	80	150	3.8	**3.1**	4.6	0.3	0.9	**0.2**	=	=	24	31	24	31	15	4
	20	160	300	944.3	**148.4**	16.2	0.8	3.1	**0.8**	=	=	45	61	45	61	20	3
	30	240	450	3222.7	**771.2**	41.8	1.9	8.6	**1.7**	=	=	70	91	70	91	22	5
	40	320	600	[a]	**5648.5**	74.9	3.7	14.1	**3.1**	589.6	=	85	121	85	121	20	4
	50	400	750	[a]	3482.5[b]	129.7	6.3	24.3	**4.9**	734.0	744.7	103	151	103	151	20	5
	60	480	900	[a]	2184.3[b]	261.6	9.1	48.8	**7.9**	877.5	869.7	146	181	146	181	28	6
	70	560	1050	[a]	6370.4[b]	345.3	14.8	71.5	**10.9**	1013.5	1011.2	155	211	155	211	23	6
	80	640	1200	6954.9[b]	2239.8[b]	517.6	20.5	100.8	**15.3**	1160.2	1162.4	169	241	169	241	25	6
	90	720	1350	[a]	3220.2[b]	748.5	27.7	144.0	**20.5**	1304.1	1305.2	190	271	190	271	27	7
	100	800	1500	[a]	3117.8[b]	1127.0	35.2	238.5	**26.7**	1447.0	1449.7	230	301	230	301	32	8

TABLE 2 Continued

Instance				Time (s)						Lower bound		# Iterations				Speedup	
				Sequential	Parallel	Sequential		Parallel		Sequential	Parallel	Sequential		Parallel			
K	*N*	RMILP	MILP	Extensive form.		Benders+ VLP	Benders+ max-flow	Benders+ VLP	Benders+ max-flow	Extensive form.		Benders+ VLP	Benders+ max-flow	Benders+ VLP	Benders+ max-flow	Sequential	Parallel
20	10	105	200	301.4	**57.8**	7.1	**0.3**	1.2	**0.3**	=	=	22	31	22	31	23	4
	20	210	400	[a]	2663.1[b]	25.1	**1.2**	5.3	**1.2**	377.2	381.9	45	61	45	61	20	4
	30	315	600	[a]	2057.8[b]	62.3	3.0	10.8	**2.2**	555.1	562.3	69	91	69	91	20	4
	40	420	800	[a]	3342.5[b]	124.5	5.1	22.0	**4.1**	732.8	751.0	91	121	91	121	24	5
	50	525	1000	[a]	5229.0[b]	210.3	8.9	38.0	**6.5**	913.2	931.5	109	151	109	151	23	5
	60	630	1200	[a]	[a]	336.4	13.9	63.6	**10.1**	1122.9	1114.0	126	181	126	181	24	6
	70	735	1400	[a]	[a]	523.9	20.9	99.5	**14.8**	1300.1	1084.7	143	211	143	211	25	6
	80	840	1600	[a]	[a]	756.3	27.9	141.9	**19.9**	1471.4	1454.6	163	241	163	241	27	7
	90	945	1800	[a]	[a]	1042.4	35.8	194.1	**25.9**	1645.4	1657.0	179	271	179	271	29	7
	100	1050	2000	[a]	[a]	1547.8	45.3	359.0	**33.1**	1847.4	1816.4	216	301	216	301	34	10
25	10	130	250	867.5	**151.6**	12.2	**0.4**	2.0	**0.4**	=	=	24	31	24	31	30	5
	20	260	500	[a]	2764.1[b]	38.6	1.6	6.3	**1.2**	461.2	462.9	43	61	43	61	24	5
	30	390	750	[a]	3584.0[b]	92.5	3.7	16.8	**2.8**	686.5	691.7	66	91	66	91	25	6
	40	520	1000	[a]	5034.6[b]	173.7	6.9	29.4	**5.2**	896.3	899.1	84	121	84	121	25	6
	50	650	1250	[a]	6221.3[b]	300.1	11.7	52.1	**8.2**	1127.0	1126.0	104	151	104	151	25	5
	60	780	1500	[a]	3483.5[b]	510.4	18.2	96.7	**12.2**	1333.2	1334.3	137	181	137	181	28	7
	70	910	1750	[a]	[a]	695.9	26,4	126,7	**17.8**	1550.9	1567.0	136	211	136	211	26	7
	80	1040	2000	[a]	[a]	1089.7	35.8	204.0	**24.1**	1787.8	1801.1	169	241	169	241	30	8
	90	1170	2250	[a]	[a]	1434.8	44.5	339.4	**31.1**	1984.0	1990.3	180	271	180	271	32	10
	100	1300	2500	[a]	[a]	2234.1	55.3	813.2	**40.1**	2240.2	2222.6	208	301	208	301	40	20

Note: Bold numbers mark best results for each group (extensive formulation and Benders). = means lower bound equals upper bound.

[a] Time limit reached (10 000 s).

[b] Premature termination (out of memory).

TABLE 3 Computational results for instance class 1 with larger graphs

Instance				Time (s)				# Iterations				Speedup	
				Sequential		Parallel		Sequential		Parallel			
K	*N*	RMILP	MILP	Benders+ VLP	Benders+ max-flow	Benders+ VLP	Benders+ max-flow	Benders+ VLP	Benders+ max-flow	Benders+ VLP	Benders+ max-flow	Sequential	Parallel
50	10	255	500	32.2	0.7	8.1	**0.6**	25	31	25	31	46	14
	20	510	1000	77.7	2.5	15.8	**1.7**	39	61	39	61	31	9
	30	765	1500	185.5	5.6	36.4	**3.8**	63	91	63	91	33	10
	40	1020	2000	367.9	18.0	94.8	**11.9**	78	121	78	121	20	8
	50	1275	2500	1010.2	32.2	160.2	**18.6**	101	151	97	151	31	9
	60	1530	3000	1416.5	51.4	365.6	**27.5**	116	181	116	181	28	13
	70	1785	3500	2334.1	76.3	764.5	**38.2**	141	211	141	211	31	20
	80	2040	4000	3183.8	98.4	1178.7	**47.9**	159	241	159	241	32	25
	90	2295	4500	4844.7	111.3	1678.7	**68.0**	198	271	198	271	44	25
	100	2550	5000	6695.0	126.6	2419.5	**86.0**	213	301	213	301	53	28
75	10	380	750	93.5	1.5	25.1	**1.1**	26	31	26	31	62	23
	20	760	1500	243.0	5.0	49.8	**3.3**	44	61	44	61	49	15
	30	1140	2250	419.9	10.8	138.1	**11.6**	60	91	60	91	39	12
	40	1520	3000	1317.0	29.8	774.4	**20.8**	70	121	70	121	44	37
	50	1900	3750	2387.0	43.8	1912.1	**32.3**	104	151	104	151	54	59
	60	2280	4500	3659.1	86.0	2051.7	**47.4**	124	181	124	181	43	43
	70	2660	5250	5245.4	126.6	3478.9	**65.6**	139	211	139	211	41	53
	80	3040	6000	7283.4	167.9	5811.4	**87.5**	168	241	168	241	43	66
	90	3420	6750	9146.4	189.8	[b]	**64.8**	174	271	–	271	48	–
	100	3800	7500	8632.4	244.9	[b]	**82.1**	206	301	–	301	35	–
100	10	505	1000	198.5	2.8	90.0	**1.7**	25	31	25	31	71	53
	20	1010	2000	425.3	8.2	196.1	**5.3**	38	61	38	61	52	37
	30	1515	3000	910.0	16.6	485.0	**10.6**	59	91	59	91	55	46
	40	2020	4000	1707.8	28.3	1136.9	**17.8**	86	121	86	121	60	64
	50	2525	5000	2480.1	44.1	1833.4	**27.9**	102	151	102	151	56	66
	60	3030	6000	3996.5	60.3	[b]	**41.4**	122	181	–	181	66	–
	70	3535	7000	[a]	183.0	[b]	**59.6**	143	211	–	211	55	–
	80	4040	8000	[a]	280.2	[b]	**135.5**	114	241	–	241	36	–
	90	4545	9000	[b]	[b]	[b]	[b]	110	198	–	–	–	–
	100	5050	10 000	[b]	[b]	[b]	[b]	–	–	–	–	–	–

Note: Bold numbers mark best results. – means quantity not computed (e.g., out of memory or out of time).

[a] Time limit reached (10 000 s).

[b] Premature termination (out of memory).

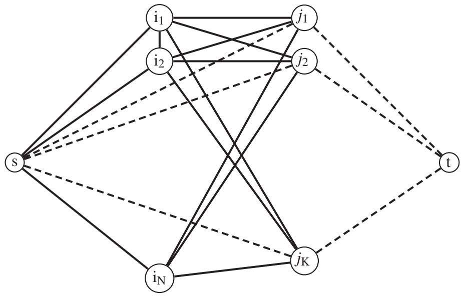

FIGURE 6 Instance class 1 with $N + K + 2$ nodes

(comparing both sequential and parallel performance); a speedup of L means that Benders+max-flow algorithm was L times faster than Benders+VLP. Thus, the greater the speedup the better and speedups >1 mean that we outperform the Benders+VLP algorithm.

For the extensive formulation, CPLEX computes early on an optimal solution for all instances of Table 2. However, the computational effort to prove its optimality increases significantly both with the number of scenarios, K, and the number of the "i"-nodes, N. Out of the 40 instances, CPLEX solves 25 instances within the time limit, sequentially or parallel. As expected, for those instances which could not be solved, we observe an increasing gap with an increase in K and N. We observe that CPLEX requires much more memory in the parallel mode compared to the sequential one. For the easier instances, the sequential algorithm tends to outperform the parallel; this changes with the harder instances.

The two Benders algorithms outperform the extensive formulation consistently. In parallel mode, both Benders algorithms are always superior for instances with 15 or 20 scenarios (these are the hard instances). For example, for the instance with 20 scenarios and $N = 100$, CPLEX (sequential) spends about 1247 seconds until the first branching with a lower bound of 1728.4 (and an upper bound of 2000) when solving the extensive formulation. At that time, Benders+max-flow would have solved the problem to proven optimality (with a gap = 0) 28 times.

The subproblems of both Benders algorithms can be naturally parallelized for the scenarios. Parallelization of the master problem does not yield significant gains because CPLEX does not parallelize so well for MILPs. Further, the run times of both parallel Benders implementations are never worse compared to the sequential one. In addition, the Benders+VLP algorithm benefits quite significantly from parallelization. We observe improvements up to six times. In contrast, the parallelization of the Benders+max-flow does not yield such significant gains. The reason is that the time spent to solve the Benders subproblems are rather small, compared to the time spent to solve the master problems.

The number of iterations required for the Benders algorithms are rather small, especially taking into account that all instances are solved to a gap of = 0. The reason that the absolute gaps is = 0, and not just a relative gap of <0.001%, is the special structure of the two-stage stochastic minimum $s - t$ cut problems as discussed at the end of Section 5. But this structure also explains the relatively fast convergence of the Benders algorithms: the lower bound improves quite significantly through the iterations. In addition, there are two other interesting observations. First, the number of iterations seems not to depend on the number of scenarios present in the instance; N determines the number of Benders iterations. This is explained by the special symmetric structure of the second-stage graphs and cost. Second, the number of iterations for Benders+max-flow remains identical for identical N. This has to do with the sequence of subproblem and master solutions computed. We analyze this further using our second instance class.

Computational results for larger graphs of instance class 1 are reported in Table 3. The organization of Table 3 is the same as Table 2, except that we no longer report the results of the extensive formulation, as the Benders decomposition implementations outperformed the extensive formulation already on smaller graphs of Table 2. We observe that the parallel Benders+max-flow algorithm converges always the fastest. The speedup of the Benders+max-flow to the vanilla Benders algorithm ranges between factor 8 and 71. The Benders+max-flow algorithm can solve instances with up to 100 scenarios and $N = 80$ in less than 300 seconds. Larger instances may have been solved, if more memory was available.

6.2 | Instance class 2

For the second instance class, consider Figure 7. There are $K \le \overline{K}$ scenarios, t_k is the terminal node for scenario $k = 1, 2, \ldots, K$, and s is the root node for all scenarios. Each of the nodes $i_1, \ldots, i_N$ is connected to each node $j_1, \ldots, j_{\overline{K}}$, and adjacent i nodes are connected to each other as well. The solid lines have a cost of $U[1, 10]$ in the first stage and ∞ in the second stage, for all scenarios; here $U[1, 10]$ is a discrete random variate generated uniformly among the 10 discrete values $\{1, 2, \ldots, 10\}$. All dashed lines have first-stage cost ∞. All dashed lines have a cost of $U[1, 10]$ in the second-stage for all scenarios, except for

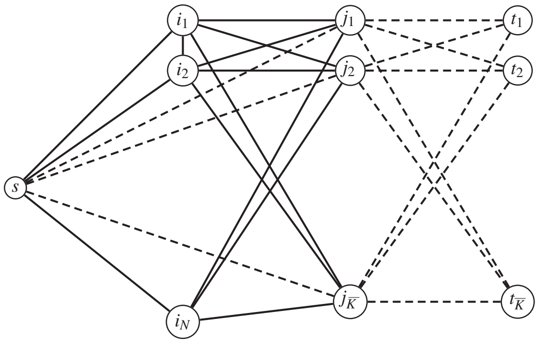

FIGURE 7 Instance class 2 with $N + 2\overline{K} + 1$ nodes

the following edges that have a cost of ∞ for scenario $k = 1, \ldots, K$: between nodes s and j_m, $m \neq k$, and between nodes j_k and t_k. All scenarios have probability $\frac{1}{K}$ and we set $\max\{100{,}000, 10(N \cdot K + 2 \cdot \overline{K})\}$ as the infinite edge cost. If all random edge costs are 1, instead of $U[1, 10]$, then an optimal first-stage solution cuts edges between the i and the j_n, $n = 1, \ldots, K$, nodes at a cost of $N \cdot K$; the optimal second-stage solution in scenario k cuts the following edges: the single edge between nodes s and j_k, the $\overline{K} - 1$ edges between nodes j_k and t_m, $m \neq k$, and the $\overline{K} - 1$ edges between nodes j_m, $m \neq k$ and t_k; the cost for scenario k is thus $2 \cdot \overline{K} - 1$. The optimal cost for this instance is $N \cdot K + 2 \cdot \overline{K} - 1$, if all random edge costs are 1.

The second instance class has the following features compared to the first instance class: (1) the terminal nodes differ for each scenario, (2) the number of scenarios K does not change the size of the graph, as long as $K \leq \overline{K}$, (3) the edge cost are no longer 0, 1 or 100 000, and (4) the integrality gap of the LP relaxation depends strongly on the ratio of K to $\overline{K}$. For each parameter combination N, $\overline{K}$ and K, we generate five instances.

Computational results for the second instance class are summarized in Table 4. The first six columns provide information about the instance. Column 4 is a counter for each group of instances, where the random costs are varied. Instance #1 has cost of 1; instances #2-#5 have random costs. Columns 7–12 report on the run times for the different methods. The obtained relative gap by the extensive formulation (sequential and parallel) and the four different Benders implementations are shown in columns 13–18. Columns 19–22 list the number of iterations of the different Benders implementations and the last two columns provide the speedups.

Consider now Table 4. For fixed number of nodes $\overline{K}$, the difficulty level of the instances increases with an increase in the number of scenarios K. This is expected. However, the level of increase in complexity is surprising. For all tested instances with $K < \overline{K}/2$ (or for $K = \overline{K}/2$ with cost 1), the LP relaxation yielded integer values. Increasing K yields harder instances while for $K = \overline{K}$, the instances are extremely difficult to solve. Instances with cost 1 tend to be easier to solve than instances with uniform discrete random cost. The Benders implementations outperform the extensive formulation only on the difficult instances. CPLEX provides especially impressive computational speeds for the extensive formulation for instances with $K < \overline{K}$. For these instances, the integrality gap is particularly small and the LP relaxation provides strong bounds. Note that the Benders algorithms benefit only indirectly from tight LP relaxations in contrast to the extensive formulation. In contrast, on the most difficult instances with $K = \overline{K}$, the Benders+max-flow algorithm outperforms the extensive formulation. Out of the 40 instances with $K = \overline{K}$, only 7 instances can be solved by the extensive formulation (sequential/parallel) while Benders+max-flow solves 33. For example, the instance with $N = 20, K = \overline{K} = 25$ and cost 1 can be solved to optimality with zero gap in 403.6 seconds by the sequential Benders+max-flow algorithm; the sequential extensive formulation has a gap of 2.7% after 10 000 seconds and the parallel mode runs of out memory after more than 2400 seconds with a gap of 3.54%.

The reported speedups in Table 4 are not that impressive compared to instance class 1. The main reason is the increase in the number of Benders iterations for Benders+max-flow compared to Benders+VLP. Assuming the same number of iterations, the Benders+max-flow algorithm outperforms the Benders+VLP by at least one order of magnitude, for all tested instances.

Overall, the computational performance of the Benders implementations compared to the extensive form is mixed. While for instance class 1 the Benders implementations yields very promising results, for instance class 2, the Benders algorithms are more challenged. In instance class 2, the "easy instances" are solved more efficiently by the extensive formulation and for the "difficult" instances, the Benders implementations run out of time or out of memory quickly. However, the computed bounds of the Benders+max-flow implementations are consistently better than the ones from the extensive form. Therefore, the results for instance class 2 are not that mature and are of preliminary nature, yet, providing promising results.

TABLE 4 Computational results for instance class 2

						Time (s)						GAP						# Iterations					
Instance						Sequential	Parallel	Sequential		Parallel		Sequential	Parallel	Sequential		Parallel		Sequential		Parallel		Speedup	
N	*K*	$\overline{K}$	#	RMILP	MILP	Extensive form.		Benders+ VLP	Benders+ max-flow	Benders+ VLP	Benders+ max-flow	Extensive form.		Benders+ VLP	Benders+ max-flow	Benders+ VLP	Benders+ max-flow	Benders+ VLP	Benders+ max-flow	Benders+ VLP	Benders+ max-flow	Sequential	Parallel
15	5	15	1	104.0	104.0	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡		
			2	403.0	628.0	**0.42**	0.69	74.60	**36.06**	67.59	37.09	=	=	=	=	=	=	40	92	40	92	2.07	1.82
			3	421.7	656.2	**0.78**	1.34	78.88	**26.49**	71.98	27.09	=	=	=	=	=	=	44	92	44	92	2.98	2.66
			4	389.7	610.2	**0.94**	1.70	134.28	75.17	123.18	**73.62**	=	=	=	=	=	=	54	108	54	108	1.79	1.67
			5	397.1	616.1	**0.73**	1.24	89.71	**35.08**	84.09	36.55	=	=	=	=	=	=	44	96	44	96	2.56	2.30
15	10	15	1	149.0	179.0	**0.45**	0.74	16.18	11.65	**7.47**	11.32	=	=	=	=	=	=	66	155	66	155	1.38	0.65
			2	813.5	975.0	**0.46**	0.79	17.45	41.22	**10.01**	41.32	=	=	=	=	=	=	69	176	69	176	0.58	0.24
			3	840.3	1017.3	**0.66**	1.01	18.62	45.45	**10.56**	46.07	=	=	=	=	=	=	68	179	68	179	0.40	0.22
			4	809.1	1021.1	**0.94**	1.35	28.45	52.92	**19.91**	54.14	=	=	=	=	=	=	81	187	81	187	0.53	0.36
			5	838.3	1008.3	**0.64**	0.99	20.44	47.50	**11.64**	46.04	=	=	=	=	=	=	75	179	75	179	0.43	0.25
15	15	15	1	149.0	254.0	**8.72**	17.71	197.57	**33.52**	163.93	34.61	=	=	=	=	=	=	214	230	214	230	5.89	4.73
			2	841.1	1406.1	629.32	**84.38**	715.60	**391.23**	677.00	396.53	=	=	=	=	=	=	296	354	296	354	1.82	1.70
			3	858.5	1439.5	388.58	**80.33**	1190.37	683.16	1164.60	**680.33**	=	=	=	=	=	=	342	391	342	391	1.74	1.71
			4	822.2	1365.2	235.13	**64.75**	631.58	245.02	597.26	**250.12**	=	=	=	=	=	=	272	306	272	306	2.58	2.39
			5	853.4	1410.9	118.69	**49.16**	364.37	**240.63**	350.17	239.85	=	=	=	=	=	=	244	303	248	303	1.51	1.45
15	20	20	1	196.5	339.0	659.97	**20.84**	578.41	**111.74**	510.73	115.70	=	=	=	=	=	=	283	305	278	305	5.18	4.41
			2	1057.4	1782.9	[b]	[b]	4361.06	740.36	1799.49	**698.69**	4.94%	4.05%	=	=	=	=	368	409	362	409	5.89	2.58
			3	1064.5	1793.0	[a]	[b]	3142.43	625.09	1242.02	**623.56**	3.52%	4.15%	=	=	=	=	369	406	353	406	5.03	1.99
			4	1088.4	1833.4	[b]	[b]	1324.43	**491.15**	978.02	493.08	4.94%	4.48%	=	=	=	=	316	379	321	379	2.70	1.98
			5	1130.2	1913.2	[b]	[b]	1566.23	591.88	1081.02	**569.90**	4.07%	4.03%	=	=	=	=	329	386	326	386	2.65	1.90
15	25	25	1	244.0	424.0	[b]	**109.57**	2188.98	240.75	1561.37	**231.31**	2.77%	=	=	=	=	=	379	379	382	379	9.09	6.75
			2	1347.7	2303.2	[a]	[b]	[a]	1449.50	5740.46	**1445.86**	6.71%	7.53%	10.67%	=	=	=	167	525	508	525	>6.90	3.97
			3	1354.6	2324.6	[b]	[b]	[a]	2394.90	[a]	**2378.87**	7.78%	8.37%	4.99%	=	4.83%	=	432	551	485	551	>4.19	>4.22
			4	1354.8	2313.8	[b]	[b]	[a]	**1933.22**	4869.00	1941.65	7.37%	6.45%	5.16%	=	=	=	478	577	487	577	>5.19	2.51
			5	1371.9	2338.4	[a]	[b]	[a]	1860.25	5911.20	**1854.18**	6.18%	6.67%	5.23%	=	=	=	436	554	503	554	>5.40	3.19
15	30	30	1	291.5	509.0	[b]	[b]	[a]	459.45	[b]	**440.70**	1.41%	4.27%	21.18%	=	21.18%	=	81	452	81	452	>21.76	–
			2	1566.0	2685.5	[b]	[b]	[a]	5225.15	[b]	**5123.10**	9.33%	8.15%	22.19%	=	18.34%	=	77	743	97	743	>1.91	–
			3	1608.5	?	[a]	[b]	[a]	[a]	[a]	[a]	9.50%	8.44%	21.61%	**4.05%**	17.88%	**4.05%**	82	769	103	771	–	–
			4	1604.5	?	[b]	[b]	[a]	[a]	[a]	[a]	9.51%	8.89%	21.18%	**4.16%**	18.89%	**4.16%**	83	695	97	695	–	–
			5	1624.9	2805.9	[b]	[b]	[a]	4226.93	[a]	**4225.88**	9.29%	8.80%	21.60%	=	16.83%	=	80	683	112	683	>2.36	>2.36
20	5	20	1	139.0	139.0	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡		
			2	733.8	733.8	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡		
			3	750.4	750.4	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡		
			4	778.6	778.6	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡		
			5	758.6	758.6	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡		
20	10	20	1	239.0	239.0	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡		
			2	1250.0	1275.5	**0.41**	1.07	**24.22**	78.17	**10.85**	80.71	=	=	=	=	=	=	68	235	68	235	0.30	0.13
			3	1247.4	1247.4	**0.27**	0.81	**23.76**	77.52	**10.68**	80.21	=	=	=	=	=	=	66	227	66	227	0.30	0.13
			4	1382.9	1406.9	**0.39**	0.98	**30.18**	104.36	**15.22**	103.95	=	=	=	=	=	=	78	248	78	248	0.28	0.14
			5	1377.9	1424.9	**0.65**	1.12	**30.01**	85.84	**14.08**	86.54	=	=	=	=	=	=	77	237	77	237	0.34	0.16

TABLE 4 Continued

Instance						Time (s)						GAP						# Iterations					
						Sequential	Parallel	Sequential		Parallel		Sequential	Parallel	Sequential		Parallel		Sequential		Parallel		Speedup	
N	K	$\overline{K}$	#	RMILP	MILP	Extensive form.		Benders+ VLP	Benders+ max-flow	Benders+ VLP	Benders+ max-flow	Extensive form.		Benders+ VLP	Benders+ max-flow	Benders+ VLP	Benders+ max-flow	Benders+ VLP	Benders+ max-flow	Benders+ VLP	Benders+ max-flow	Sequential	Parallel
20	15	20	1	249.0	339.0	**4.10**	5.62	105.78	**64.05**	68.83	65.52	=	=	=	=	=	=	122	306	122	306	1.65	1.05
			2	1358.7	1856.7	**5.35**	5.95	165.85	239.00	**128.37**	237.69	=	=	=	=	=	=	124	337	124	337	0.69	0.54
			3	1382.6	1895.6	**6.97**	8.42	201.98	256.41	**165.50**	263.27	=	=	=	=	=	=	139	343	139	343	0.78	0.62
			4	1328.5	1825.5	**10.15**	11.37	184.09	241.22	**145.99**	245.01	=	=	=	=	=	=	136	330	136	330	0.76	0.59
			5	1338.2	1835.7	**5.63**	6.49	193.91	229.57	**157.36**	234.35	=	=	=	=	=	=	129	335	129	335	0.84	0.67
20	20	20	1	249.0	439.0	2.30600	[b]	970.12	**158.37**	884.42	183.57	=	2.59%	=	=	=	=	359	404	359	403	6.01	4.81
			2	1332.9	2286.5	[b]	[b]	9653.78	1470.02	4343.13	**1143.53**	5.74%	4.38%	=	=	=	=	442	534	461	524	6.56	3.79
			3	1330.6	2294.6	[b]	[b]	[a]	4649.08	6487.17	**1976.00**	6.07%	4.71%	6.64%	=	=	=	425	646	538	627	>2.15	3.28
			4	1313.4	2262.9	[b]	[b]	[a]	**3383.51**	9630.69	3406.10	6.11%	5.01%	6.50%	=	=	=	401	563	454	563	>2.95	2.82
			5	1352.1	?	[b]	[b]	[a]	[a]	[a]	[a]	6.45%	6.49%	6.42%	5.94%	6.10%	5.94%	404	609	461	609	–	–
20	25	25	1	309.0	549.0	[a]	[b]	[a]	**403.55**	**5777.69**	406.01	2.70%	3.54%	13.71%	=	=	=	177	505	475	505	>24.78	14.23
			2	1689.2	2941.7	[b]	[b]	[a]	3894.27	[a]	**3882.99**	7.92%	7.58%	16.57%	=	15.86%	=	143	673	151	673	>2.56	>2.57
			3	1761.4	?	[b]	[b]	[a]	[a]	[a]	[a]	8.19%	8.30%	19.84%	**5.01%**	15.44%	**5.01%**	117	741	155	737	–	–
			4	1728.2	?	[b]	[b]	[a]	[a]	[a]	[a]	8.68%	7.73%	19.97%	**5.05%**	6.33%	**5.05%**	116	662	345	663	–	–
			5	1682.0	?	[b]	[b]	[a]	[a]	[a]	[a]	8.31%	8.10%	17.25%	**5.09%**	5.38%	**5.09%**	138	712	477	712	–	–
25	5	25	1	149.0	149.0	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡		
			2	845.0	845.0	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡		
			3	840.2	840.2	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡		
			4	812.6	812.6	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡		
			5	806.0	806.0	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡		
25	10	25	1	249.0	249.0	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡		
			2	1237.1	1237.1	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡		
			3	1405.7	1405.7	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡		
			4	1350.1	1350.1	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡		
			5	1397.3	1397.3	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡		
25	15	25	1	349.0	349.0	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡		
			2	1671.4	1882.4	**1.85**	3.28	75.07	311.00	**39.31**	307.68	=	=	=	=	=	=	83	344	84	344	0.24	0.12
			3	1705.5	2012.5	**3.23**	3.98	93.53	314.51	**59.48**	312.94	=	=	=	=	=	=	95	342	94	342	0.29	0.19
			4	1683.5	1852.0	**1.75**	3.23	77.59	304.23	**37.34**	304.16	=	=	=	=	=	=	78	342	79	342	0.25	0.12
			5	1648.0	1867.5	**2.15**	3.67	85.18	365.77	**39.44**	363.36	=	=	=	=	=	=	87	369	83	369	0.23	0.10
25	25	25	1	374.0	674.0	[b]	[b]	[a]	**649.06**	[b]	673.29	5.23%	4.72%	23.18%	=	21.60%	=	118	679	133	630	>15.40	–
			2	2053.6	3622.6	[b]	[b]	[a]	5988.01	[b]	5781.40	8.73%	7.95%	23.19%	=	22.07%	=	123	828	130	828	>1.67	–
			3	2052.4	?	[b]	[b]	[a]	[a]	[a]	[a]	9.28%	8.34%	22.96%	**5.45%**	19.67%	**5.45%**	125	750	149	750	–	–
			4	2080.8	3685.8	[b]	[b]	[a]	4317.81	[b]	**4301.02**	8.89%	8.88%	23.93%	=	22.88%	=	116	794	125	794	>2.31	–
			5	2062.8	3649.8	[b]	[b]	[a]	4825.03	[a]	**4821.85**	8.03%	8.07%	23.84%	=	20.44%	=	117	812	142	812	>2.07	–

Note: Bold numbers mark best results for each group (extensive formulation and Benders). = means lower bound equals upper bound. – indicates quantity not computed (e.g., out of memory or out of time). ≡ indicates LP relaxation yields integer solution. ? indicates optimal objective function value unknown.

[a] Time limit reached (10 000 s).

[b] Premature termination (out of memory).

7 | CONCLUSIONS

Based on the classical minimum $s-t$ cut problem, we introduce the two-stage stochastic minimum $s-t$ cut problem. We provide a MILP formulation which is an extension of the standard linear 0-1 programming model for the deterministic minimum $s-t$ cut problem. We prove that the constraint matrix of the new formulation loses its total unimodularity property, in general; however, the matrix preserves the property if the considered graph is a tree. This fact turns out to be not surprising as we show that similar to many other stochastic extensions of classical combinatorial optimization problems (e.g., minimum spanning tree [13]), the arc-based version of the two-stage stochastic minimum $s-t$ cut problem is $\mathcal{NP}$-hard. In the case of trees, the two-stage stochastic minimum $s-t$ cut problem is polynomially solvable due to the total unimodularity property; we also describe a simple linear time solution algorithm. We apply Benders decomposition to the two-stage stochastic minimum $s-t$ cut problem. Instead of solving the Benders subproblems as LPs, we present an efficient implementation which exploits the special structure of the subproblems and uses a maximum-flow algorithm to compute the Benders optimality cuts. This algorithm yields promising computational results.

ACKNOWLEDGMENTS

S.R. thanks Klaus Truemper for his discussions on Section 2.2. We thank Ömer Özümerzifon (KIT) for his work on the C++ implementations. We also thank three anonymous reviewers and the associate editor for their detailed and constructive comments. O.A.P. was partially supported by the U.S. Air Force Office of Scientific Research Grants FA9550-08-1-0268 and FA9550-11-1-0037. B.S. thanks Sandia National Laboratories, USA for the part of his time spent there as a postdoctoral appointee.

ORCID

Steffen Rebennack https://orcid.org/0000-0002-8501-2785
Oleg A. Prokopyev https://orcid.org/0000-0003-2888-8630

REFERENCES

[1] S. Ahmed, *Convexity and decomposition of mean-risk stochastic programs*, Math. Program. **106** (2006), 433–446.
[2] R. Ahuja, T. Magnanti, and J. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, Englewood Cliffs, NJ, 1993.
[3] J. Benders, *Partitioning procedures for solving mixed-variables programming problems*, Numer. Math. **4** (1962), 238–252.
[4] J. Birge and F. Louveaux, *Introduction to Stochastic Programming*, Springer, New York, 1997.
[5] M. Carey and C. Hendrickson, *Bounds on expected performance of networks with links subject to failure*, Networks **14** (1984), 439–456.
[6] T. Christof and A. Löbel, 2009. Polyhedron representation transformation algorithm. http://comopt.ifi.uni-heidelberg.de/software/PORTA/index.html
[7] C. Colbourn, *The Combinatorics of Network Reliability*, Oxford University Press, New York, 1987.
[8] E. Dahlhaus, D. Johnson, C. Papadimitriou, P. Seymour, and M. Yannakakis, *The complexity of multiterminal cuts*, SIAM J. Comput. **23** (1994), 864–894.
[9] K. Dhamdhere, V. Goyal, R. Ravi, and M. Singh, *How to pay, come what may: Approximation algorithms for demand-robust covering problems*, FOCS 2005. 46th Annual IEEE Symposium on Foundations of Computer Science, 2005, pp. 367–376.
[10] K. Dhamdhere, R. Ravi, and M. Singh, *On two-stage stochastic minimum spanning trees*, Proceedings of the 11th International Conference on Integer Programming and Combinatorial Optimization, 2005, pp. 321–334.
[11] L.R. Ford and D.R. Fulkerson, *Maximal flow through a network*, Can. J. Math. **8** (1956), 399–404.
[12] L.R. Ford and D.R. Fulkerson, *Flows in Networks*, Princeton University Press, Princeton, NJ, 1962.
[13] A. Frieze, A. Flaxman, and M. Krivelevich, *On the random 2-stage minimum spanning tree*, Random Structures Algorithms **28** (2006), 24–36.
[14] D. Golovin, V. Goyal, V. Polishchuk, R. Ravi, and M. Sysikaski, *Improved approximations for two-stage min-cut and shortest path problems under uncertainty*, Math. Program. **149** (2015), 167–194.
[15] D. Golovin, V. Goyal, and R. Ravi, "*Pay today for a rainy day: Improved approximation algorithms for demand-robust min-cut and shortest path problems*," *STACS 2006, Lecture Notes in Computer Science*Springer, Berlin/Heidelberg, 2006, pp. 206–217.
[16] A. Gupta, R. Ravi, and A. Sinha, *LP rounding approximation algorithms for stochastic network design*, Math. Oper. Res. **32** (2007), 345–364.
[17] T. Harris and F. Ross, Fundamentals of a method for evaluating rail net capacities, Tech. report, Rand Corp, Santa Monica CA, 1955.
[18] K. Hastings and D. Shier, *Algebraic methods for stochastic minimum cut and maximum flow problems*, Proceedings of the 5th International Conference on Network Optimization, 2011, pp. 295–308.
[19] M. Henzinger, A. Noe, C. Schulz, and D. Strash, *Practical minimum cut algorithms*, J. Exp. Algorithmics (JEA) **23** (2018), 1–8.
[20] D. Karger, *A randomized fully polynomial time approximation scheme for the all-terminal network reliability problem*, SIAM Rev. **43** (2001), 499–522.
[21] D. Karger and C. Stein, *A new approach to the minimum cut problem*, J. ACM **43** (1996), 601–640.
[22] R. Khandekar, G. Kortsarz, V. Mirrokni, and M. Salavatipour, *Two-stage robust network design with exponential scenarios*, Algorithmica **65** (2013), 391–408.
[23] N. Kong and A. Schaefer, *A factor 1/2 approximation algorithm for two-stage stochastic matching problems*, Eur. J. Oper. Res. **172** (2006), 740–746.

[24] T. Lohmann and S. Rebennack, *Tailored Benders decomposition for a long-term power expansion model with short-term demand response*, Manage. Sci. **63** (2017), 2027–2048.
[25] J.C. Picard and M. Queyranne, *Selected applications of minimum cuts in networks*, INFOR: Inf. Syst. Oper. Res. **20** (1982), 394–422.
[26] R. Rahmaniani, T. Crainic, M. Gendreau, and W. Rei, *The Benders decomposition algorithm: A literature review*, Eur. J. Oper. Res. **259** (2017), 801–817.
[27] S. Rebennack, *Combining sampling-based and scenario-based nested Benders decomposition methods: Application to stochastic dual dynamic programming*, Math. Program. **156** (2016), 343–389.
[28] A. Ruszczyński and A. Shapiro (eds), *Handbooks in OR & MS: Stochastic Programming*, vol. **10**, Elsevier, Amsterdam, 2003.
[29] A. Schrijver, *On the history of the transportation and maximum flow problems*, Math. Program. **91** (2002), 437–445.
[30] R. Schultz, *On structure and stability in stochastic programs with random technology matrix and complete integer recourse*, Math. Program. **70** (1995), 73–89.
[31] R. Slyke and R.B. Wets, *L-shaped linear programs with applications to control and stochastic programming*, SIAM J. Appl. Math. **17** (1969), 638–663.
[32] Y. Soun and K. Truemper, *Single commodity representation of multicommodity networks*, SIAM J. Algebr. Discrete Methods **1** (1980), 348–358.
[33] R. Tahmasbi, E. Nasrabadi, and S. Hashemi, *The value of information in stochastic maximum flow problems*, Comput. Oper. Res. **40** (2013), 1744–1751.
[34] K. Truemper, *Unimodular matrices of flow problems with additional constraints*, Networks **7** (1977), 343–358.
[35] K. Truemper, *Matroid Decomposition*, Academic Press, Boston, revised edition Leibniz, Plano, TX, 1998.
[36] S. Wallace and W. Ziemba (eds), *Applications of Stochastic Programming*, SIAM, Philadelphia, PA, USA, 2005.
[37] L. Wolsey, *Integer Programming*, John Wiley & Sons, New York, 1998.

How to cite this article: Rebennack S, Prokopyev OA, Singh B. Two-stage stochastic minimum $s-t$ cut problems: Formulations, complexity and decomposition algorithms. *Networks.* 2019;1–24. https://doi.org/10.1002/net.21922