Detecting Resilient Structures in Stochastic Networks: A Two-Stage Stochastic Optimization Approach

Maciej Rysz
National Research Council, Air Force Research Laboratory, 101 West Eglin Blvd, Eglin AFB, Florida 32542

Pavlo A. Krokhmal
Department of Systems & Industrial Engineering, University of Arizona, 1127 E James E. Rogers Way, Tucson, Arizona 85721

Eduardo L. Pasiliao
Air Force Research Laboratory, 101 West Eglin Blvd, Eglin AFB, Florida 32542

We propose a two-stage stochastic programming framework for designing or identifying "resilient," or "reparable" structures in graphs whose topology may undergo a stochastic transformation. The reparability of a subgraph satisfying a given property is defined in terms of a budget constraint, which allows for a prescribed number of vertices to be added to or removed from the subgraph so as to restore its structural properties after the observation of random changes to the graph's set of edges. A two-stage stochastic programming model is formulated and is shown to be $\mathcal{NP}$ -complete for a broad range of graph-theoretical properties that the resilient subgraph is required to satisfy. A general combinatorial branch-and-bound algorithm is developed, and its computational performance is illustrated on the example of a two-stage stochastic maximum clique problem. © 2016 Wiley Periodicals, Inc. NETWORKS, Vol. 000(00), 000–000 2016

Keywords: maximum subgraph problem; stochastic graphs; resilience of subgraphs; two-stage stochastic optimization; combinatorial branch-and-bound algorithm; stochastic maximum clique problem

Received November 2016; revised November 2016; accepted December 2016

Correspondence to: P. A. Krokhmal krokhmal@email.arizona.edu; e-mail: krokhmal@email.arizona.edu

Contract grant sponsor: AFOSR grant; Contract grant number: FA9550-12-1-0142
Contract grant sponsor: DTRA grant; Contract grant number: HDTRA1-14-1-0065
Contract grant sponsor: U.S. Department of Air Force grant; Contract grant number: FA8651-14-2-0003
DOI 10.1002/net.21727
Published online in Wiley Online Library (wileyonlinelibrary.com).

1. INTRODUCTION AND MOTIVATION

An important feature to incorporate in a networked system's design is an inherent resilience to withstand random structural changes that affect the relationship characteristics between its components. A reliable system should, therefore, possess a high tolerance against a broad range of possible (failure) scenarios, and, moreover, be constructed in such a way that its properties can be restored within available resource limits.

In the present study, we pursue an approach that regards a distributed subsystem, or subgraph, to be *resilient* if it can be "*repaired*" at a minimum (or fixed) cost after a random change in the underlying graph's topology. More specifically, many graph-theoretical and network optimization problems consist in finding a subgraph with prescribed properties that has the largest (respectively, smallest) size, weight, and so on. Well-known examples include the shortest path problem, maximum clique/independent set problem, minimum vertex cover problem, and so on. In situations when the topology of the underlying graph or network may be subject to changes (e.g., deletions of vertices and/or edges), the "resilience" of the selected subgraph is often of interest. A large body of literature has been accumulated on this subject, where various interpretations of "reliability," "resilience," or "robustness" of subgraphs have been explored (see among others [13, 15, 26, 29, 32]). Typically, robustness in this context is associated with the ability of the selected subgraph to satisfy (exactly or to a certain degree) a given property, or perform a given function, and so forth, after deletion of edges and/or vertices. Several examples include network flow control, preservation of vertex and edge connectivity, maximization of overall algebraic connectivity, and prevention of catastrophic cascade failures [6–8, 15].

In this work, we adopt the point of view that a structure in a network or graph is "resilient" if it is "reparable" with respect to randomized changes in the graph's topology. Namely, we consider a general framework which assumes that a given (original) graph may undergo randomized changes in the form of edge failures or formations. It is then of interest to identify a subset of vertices in the original graph whose (user-defined) structural property can be repaired after the edge modifications take place. In each random outcome, the repairs are made by removing and/or adding vertices to the selected subset, resulting in a "repaired" subset. We additionally impose the following requirements:

i. the number of repairs made to the subset selected from the original graph is within a prescribed limit
ii. the size of the selected subset and the expected size of the repaired subsets should be as large as possible

In other words, the problem is to identify the largest possible set of vertices whose structural property can be repaired within a fixed budget, such that the expected size of the resulting subgraphs is also as large as possible.

The described concept has obvious interpretations in, for example, the defense domain, where one may be interested in identifying the largest networked or distributed system that can maintain its structure—with, perhaps, necessary repairs—under adversarial attacks. Alternatively, consider the problem of harvesting information from social or communications networks by means of planting sensory devices or recruiting informers, where a link between two sources of information indicates that the provided information may be highly correlated or duplicated. In this context, one may be interested in the largest possible *independent set* of sources, so as to obtain maximally independent and diversified information. We also want this independent set to be *reparable*, if it turns out that some relations, or network properties were unknown to us or have changed over time. In the financial setting, one may be interested in identifying the largest subset of banks that are not connected via borrowing/lending contracts, and thus are less susceptible to the "domino effect" in the event of a market crash. It is likewise desirable to have this set reparable in the event new credit links are established between banks, and so on.

Mathematically, the outlined framework lends itself naturally to the context of two-stage stochastic optimization [5, 23], which models the decision making process in the presence of uncertainties that involves two sequential decisions. The *first-stage* decision is made before the actual realization of uncertain factors can be observed. The *second-stage*, or *recourse*, decision is made on observing the realization of uncertainties, and takes into account both the preceding first-stage decision and the observed realization of stochastic parameters.

Stochastic recourse problems have gained much attention in the network literature due to their versatility for modeling uncertainties. Particular emphasis has been placed on network problems with random elements evidenced in forms that influence the overall flow distribution, demands, and costs. A number of applications examine stochastic factors in the context of vehicle routing and network flow problems where uncertainties are attributed to arc capacities or node demands (see e.g., [3, 9, 16, 18, 31]). Several similar considerations utilized a two-stage recourse framework to enhance the design of stochastic supply chain networks and network resource allocation [11, 28]. Other studies examined the preservation of connections between vertices when the edge costs are uncertain [7, 19], as well as decision making in routing problems with stochastic edge failures [30].

Although uncertainty in the aforementioned studies mostly influenced decisions related to directed flows and routing, less focus has been put on developing two-stage recourse constructs for designing/identifying graphs that are adept at maintaining their connection properties in situations when random factors affect/alter/damage their original physical characteristics. A notable non-recourse problem of finding the largest subset of vertices that form a clique with a specified probability, given that edges in the graph can fail with some probabilities, was studied in [20]. A similar approach in application to certain clique relaxations was pursued in [35]. In this work, we introduce a two-stage stochastic recourse framework for identifying "sustainable" subgraphs whose structural properties are influenced by definite edge failures and/or construction in each random scenario realization. The proposed model is general and can be adapted to address a broad range of structural graph properties, along with uncertainties in the form of vertex failures.

The remainder of the article is organized as follows. In section 2, we discuss the deterministic graph-theoretic underpinnings and establish a mathematical programming representation of the *two-stage stochastic recourse maximum subgraph problem*. Section 3 presents an efficient graph-based (combinatorial) branch-and-bound solution algorithm for instances when the desired subgraphs possess hereditary structural properties. Finally, section 4 considers a numerical case study demonstrating the effectiveness of the proposed algorithm for solving two-stage stochastic recourse maximum clique (i.e., complete graph) problems.

2. PROBLEM DEFINITION

In this section, we present a formal graph-theoretical description of the discussed framework. Before introducing the stochastic model that represents the focus of the present work, we outline the relevant deterministic concepts, which pertain to problems involving the identification of the largest subgraph/subset of a system's vertices that collectively possess a specified structural property.

2.1. Deterministic Maximum Subgraph Problem

Let $G = (V, E)$ represent an undirected graph where each vertex $i \in V$ is a component of the networked system, and an edge $(i, j) \in E$ defines a connection/relation between vertices i and j. Then, the problem of finding the largest (sub)graph

$S \subseteq V$ of vertices with a prescribed structural property Π, also known as the *maximum subgraph problem*, or *maximum* Π *problem*, is given by

$$\max_{S \subseteq V} \{|S| : G[S] \ni \Pi\}, \tag{1}$$

where $G[S]$ denotes the subgraph of G *induced* by S, that is, a graph such that any of its vertices i, j are connected by an edge if and only if (i, j) is an edge in graph G. Here and throughout the text, the relation $G[S] \ni \Pi$ stands for "$G[S]$ satisfies property Π" (we also say that S *is a* Π*-subgraph of* G); similarly, $G[S] \not\ni \Pi$ represents the negated statement.

In the context of the maximum subgraph problem (1), an important class of graph-theoretical properties Π is represented by *properties that are hereditary with respect to induced subgraphs* (or just *hereditary* for short):

Definition 1 ([1, 4, 34])**.** *Property* Π *is called hereditary with respect to induced subgraphs if for any graph that satisfies* Π*, the removal of any vertex from this graph results in an induced subgraph that also satisfies* Π*.*

The class of hereditary properties encompasses many well-known and important graph-theoretical properties, such as *completeness, independence, planarity*, and so on.

The practical and theoretical significance of hereditary properties in relation to the maximum subgraph problem stems from the fact that a large number of important and difficult graph-theoretical problems are special cases of (1) when Π is hereditary and "meaningful" in the context of the following definition.

Definition 2 ([34])**.** *Property* Π *is called nontrivial if it is satisfied by a single-vertex graph yet not satisfied by every graph, and is called interesting if the order of graphs satisfying* Π *is unbounded.*

Then, the following fundamental observation regarding problem (1) holds:

Theorem 1 (Yannakakis [34])**.** *If property* Π *is hereditary with respect to induced subgraphs, nontrivial, and interesting, then the maximum subgraph problem (1) is* $\mathcal{NP}$*-complete.*

In view of the above result, in this study we assume that the property Π is hereditary, nontrivial, and interesting.

In many practical applications, the topology of graph G in the maximum subgraph problem (1) may not always be assumed constant, and is subject to unpredictable, or stochastic changes (e.g., edge and/or vertex failures). In the next section, we present a mathematical framework for finding maximum subgraphs in the presence of such stochastic changes.

2.2. A Two-Stage Stochastic Maximum Subgraph Problem

If graph G is assumed to be stochastic, the maximum subgraph problem does not provide a guarantee or conditions under which the selected subgraph $G[S]$ satisfies the sought property Π. Therefore, in the presence of uncertainties, formulation (1) has to be modified to explicitly specify the conditions under which its solution can be considered a Π-subgraph of (stochastic) graph G. One common approach in the literature is to require that the solution of an optimization problem with stochastic data satisfies the required properties with a prescribed probability. An application of this approach to a maximum clique problem on stochastic graphs was considered in [20].

In the present endeavor, we require that the solution of the maximum subgraph problem on a stochastic graph is "reparable" in some sense. Particularly, we introduce an approach for identifying resilient maximum Π-subgraphs in situations when the topology of the underlying graph G may be subject to uncertain (random) future changes that is based on two-stage stochastic programming and which was tentatively outlined in section 1.

Given a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, where Ω is the set of random events, $\mathcal{F}$ is the sigma-algebra, and $\mathbb{P}$ is the probability measure, we assume that the topology of a graph $G = (V, E)$ may undergo a random transformation at some moment in the future, resulting in an updated graph $G(\omega) = (V, E(\omega))$, $\omega \in \Omega$. In this work, it is assumed for simplicity that only the set of edges $E = E(\omega)$ may be dependent on the random event ω, while the set of vertices V is constant. As it will be seen next, the proposed formulation and solution method can be generalized to account for possibility of a stochastic set V.

We also employ an assumption prevalent in stochastic programming literature, that the set Ω is finite, $\Omega = \{\omega_1, \ldots, \omega_N\}$, with $\mathbb{P}(\omega_k) = p_k > 0$ for $k = 1, \ldots, N$, and $\sum_k p_k = 1$. Consequently, the possible changes to the topology of graph G are observed in the form of N discrete scenarios $\{G(\omega_1), \ldots, G(\omega_N)\}$, where $G(\omega_k) = (V, E(\omega_k))$. For notational convenience, we will denote $G_k = G(\omega_k)$, $E_k = E(\omega_k)$. Also, to emphasize that the original graph G represents the unchanged, or "initial" state of the distributed system, we denote $G_0 = G = (V, E_0)$, where $E_0 = E$ represents the initial set of edges in the graph.

Characterization of "resilient" substructures in graphs subjected to randomized topology changes via the formalism of two-stage stochastic programming is the key feature of the proposed approach. In general, a two-stage stochastic programming model may be presented in the form

$$\begin{aligned} &\min\{f_1(\mathbf{x}) + \mathbb{E} f_2(\mathbf{x}, \mathbf{y}(\omega), \omega) : \\ &\quad \mathbf{h}_1(\mathbf{x}) \leq \mathbf{0},\ \mathbf{h}_2(\mathbf{x}, \mathbf{y}(\omega), \omega) \leq \mathbf{0},\ \omega \in \Omega\}. \end{aligned} \tag{2}$$

Here, $\mathbf{x}$ represents the *first-stage* decision/action that is made before the actual realization of the uncertain event ω can be observed. Associated with the first-stage decision are the first-stage cost $f_1(\mathbf{x})$ and the first-stage constraints $\mathbf{h}_1(\mathbf{x}) \leq \mathbf{0}$.

Since the first-stage decision $\mathbf{x}$ may not be optimal for every possible realization of ω, a *second-stage*, or *recourse*, decision $\mathbf{y} = \mathbf{y}(\omega)$ is made after the actual realization of ω has been observed, such that the second-stage cost $f_2(\mathbf{x}, \mathbf{y}(\omega), \omega)$ is minimized. The recourse decision $\mathbf{y}(\omega)$ must also satisfy the second-stage constraints $\mathbf{h}_2(\mathbf{x}, \mathbf{y}(\omega), \omega) \leq \mathbf{0}$ for any given first-stage $\mathbf{x}$. Importantly, the second-stage decision depends explicitly on the specific realization of ω as well as on the first-stage decision $\mathbf{x}$. In turn, the first-stage decision must take into account all possible realizations of ω and the subsequent recourse decisions $\mathbf{y}(\omega)$. This interdependency is emphasized by the following "nested," or *recourse*, representation of the extensive form (2):

$$\min \{f_1(\mathbf{x}) + \mathbb{E}Q(\mathbf{x}, \omega) : \mathbf{h}_1(\mathbf{x}) \leq \mathbf{0}\}, \tag{3a}$$

$$\text{where } Q(\mathbf{x}, \omega) = \min \{f_2(\mathbf{x}, \mathbf{y}(\omega), \omega) : \mathbf{h}_2(\mathbf{x}, \mathbf{y}(\omega), \omega) \leq \mathbf{0}\}. \tag{3b}$$

According to the above, the following two-stage framework is adopted for identification of "resilient" Π-subgraphs in G_0:

First stage: Given a graph $G_0 = (V, E_0)$, find a set of vertices $S_0 \subseteq V$ such that the induced subgraph $G_0[S_0]$ satisfies Π.

Observation of uncertainty: Graph G_0 undergoes a randomized change of topology. It is assumed that the resulting graph $G_k = (V, E_k)$ is chosen at random with probability p_k from a collection of graphs $\{G_1, \ldots, G_N\}$.

Second stage: For any given realization G_k, select sets $\Delta_k^+ \subseteq V \setminus S_0$ and $\Delta_k^- \subseteq S_0$, such that after "augmentation" or "repair" of the original set S_0, the resulting set S_k,

$$S_k := (S_0 \setminus \Delta_k^-) \cup \Delta_k^+,$$

induces a subgraph $G_k[S_k]$ on G_k that satisfies Π.

Objective and reparability: Sets S_0 and $\Delta_k^{\pm}$ (equivalently, S_k) must be chosen in such a way that the expected size of a Π-subgraph in the first and second stages is maximized, and the sets $\Delta_k^{\pm}$ contain no more than M vertices,

$$|\Delta_k^+| + |\Delta_k^-| \leq M. \tag{4}$$

Then, the *two-stage stochastic maximum subgraph (TSMS) problem* can be stated in the graph-theoretical form as follows:

$$\max \quad |S_0| + \sum_{k \in \mathcal{N}} p_k |S_k| \tag{5a}$$

$$\text{s.t.} \quad G_k[S_k] \ni \Pi, \quad \forall k \in \{0\} \cup \mathcal{N} \tag{5b}$$

$$|S_0 \setminus S_k| + |S_k \setminus S_0| \leq M, \quad \forall k \in \mathcal{N} \tag{5c}$$

$$S_k \subseteq V, \quad \forall k \in \{0\} \cup \mathcal{N}, \tag{5d}$$

where $\mathcal{N} = \{1, \ldots, N\}$. Obviously, the defined above delta-sets $\Delta_k^{\pm}$ are related to the second-stage sets S_k as

$$\Delta_k^+ = S_k \setminus S_0, \quad \Delta_k^- = S_0 \setminus S_k, \quad k \in \mathcal{N}.$$

The extended formulation (5) of the two-stage stochastic maximum subgraph problem can be presented in the recourse form similar to (3):

$$\max_{S_0 \subseteq V} \left\{ |S_0| + \sum_{k \in \mathcal{N}} p_k Q_k(S_0) : G_0[S_0] \ni \Pi \right\}, \tag{6a}$$

where the second-stage function Q_k has the form

$$Q_k(S) = \max_{S_k \subseteq V} \{|S_k| : G_k[S_k] \ni \Pi,\ |S \setminus S_k| + |S_k \setminus S| \leq M\}. \tag{6b}$$

2.2.1. A Mathematical Programming Formulation of the TSMS Problem A mathematical programming formulation of the TSMS problem can be obtained in a straightforward way by introducing a binary vector $\mathbf{x} \in \{0,1\}^{|V|}$ such that $x_i = 1$ if $i \in S_0$ and $x_i = 0$ otherwise, and, similarly, vectors $\mathbf{y}_k \in \{0,1\}^{|V|}$, $k \in \mathcal{N}$, to indicate whether vertex $i \in V$ belongs to the subset S_k in the second stage. Further, let $\mathbf{\Pi}_G(\mathbf{x}) \leq \mathbf{0}$ be the "structural" constraints associated with the property Π; namely, $\mathbf{\Pi}_G(\mathbf{x}) \leq \mathbf{0}$ if and only if $G[S_0]$, where $S_0 = \{i \in V : x_i = 1\}$, satisfies Π. Similarly, second-stage sets $S_k = \{i \in V : y_{ik} = 1\}$, $k \in \mathcal{N}$, represent induced Π-subgraphs in G_k if and only if $\mathbf{\Pi}_{G_k}(\mathbf{y}_k) \leq \mathbf{0}$, $k \in \mathcal{N}$. Obviously, the functional form $\mathbf{\Pi}_G(\cdot) \leq \mathbf{0}$ of the structural constraints may not be unique for any given property Π.

The corresponding 0-1 integer programming formulation of TSMS problem (5) then takes the form

$$\max \quad \mathbf{1}^\top \mathbf{x} + \sum_{k \in \mathcal{N}} p_k \mathbf{1}^\top \mathbf{y}_k \tag{7a}$$

$$\text{s.t.} \quad \mathbf{\Pi}_{G_0}(\mathbf{x}) \leq \mathbf{0} \tag{7b}$$

$$\mathbf{\Pi}_{G_k}(\mathbf{y}_k) \leq \mathbf{0}, \quad \forall k \in \mathcal{N} \tag{7c}$$

$$\|\mathbf{x} - \mathbf{y}_k\|_1 \leq M, \quad \forall k \in \mathcal{N} \tag{7d}$$

$$\mathbf{x}, \mathbf{y}_k \in \{0,1\}^{|V|}, \quad \forall k \in \mathcal{N}, \tag{7e}$$

where $\mathbf{1}$ denotes the vector of ones of an appropriate dimension. Constraints (7d) impose the previously described budgetary restrictions.

2.2.2. Computational Complexity. Complexity of the two-stage stochastic maximum subgraph problem (5)–(6) is established in the next two propositions. Let $V(G)$ represent the set of vertices in the graph G, and consider the following decision version of the two-stage stochastic maximum subgraph problem (5)–(6):

Decision problem $\langle (G_0, \ldots, G_N), (p_1, \ldots, p_N), M, q \rangle$: Given a set of $N+1$ graphs $G_0, \ldots, G_N$ such that $V(G_0) = \cdots = V(G_N)$, a set of positive rational numbers $p_1, \ldots, p_N$ such that $p_1 + \cdots + p_N = 1$, an integer $M \geq 0$, and a rational $q \geq 0$, determine whether graphs G_i contain Π-subgraphs S_i such that $|S_0 \setminus S_i| + |S_i \setminus S_0| \leq M$ for all $i = 1, \ldots, N$, and $|S_0| + \sum_{i=1}^N p_i |S_k| \geq q$

Similarly, the decision version of the maximum subgraph problem is as follows:

Decision problem $\langle G, m\rangle$: Given a graph G and a nonnegative integer m, determine whether G contains a Π-subgraph S such that $|S| \geq m$

Proposition 1. *The decision version of the two-stage stochastic maximum subgraph problem (5) is $\mathcal{NP}$ -complete, provided that the corresponding maximum subgraph problem is $\mathcal{NP}$ -complete.*

Proof. Noting that the two-stage stochastic maximum subgraph problem is obviously in $\mathcal{NP}$, we prove its $\mathcal{NP}$-completeness by reduction from the maximum subgraph problem. Given an instance $\langle G, m\rangle$ of the maximum subgraph problem, let $N = 1$, $p_1^* = 1$, $G_0^* = G_1^* = G$, $q^* = 2m$, and select an arbitrary integer $M^* \geq 0$. Then, sets $S_0^* = S_1^* \subseteq V(G_i^*)$ satisfy the conditions $|S_0^* \setminus S_1^*| + |S_1^* \setminus S_0^*| \leq M^*$ and $|S_0^*| + \sum_{i=1}^{N} p_i^* |S_i^*| \geq m + m = q^*$ if and only if there exists $S \subseteq V(G)$ of order $|S| \geq m$. ∎

Next, we observe that for any given first-stage solution, "repairing" it in the second stage via solving the second-stage problem (6b) is $\mathcal{NP}$-complete as well. To this end, the corresponding decision version $\langle G_k, S, M, q\rangle$ of the second-stage maximum subgraph problem is formulated as follows: given a second-stage graph G_k, a first-stage solution $S \subseteq V(G_0) = V(G_k)$, and integer numbers $M \geq 0$ and $q \geq 0$, determine if a Π-subgraph $S_k \subseteq V(G_k)$ of order at least q exists such that $|S \setminus S_k| + |S_k \setminus S| \leq M$. Then, the next observation holds.

Proposition 2. *The decision version of the second-stage maximum subgraph problem problem (6b) at any scenario $k \in \mathcal{N}$ is $\mathcal{NP}$ -complete if property Π is such that the maximum subgraph problem is $\mathcal{NP}$ -complete.*

Proof. Note that the second-stage maximum subgraph problem is in $\mathcal{NP}$. Next, observe that the order of a Π-subgraph of G_k that satisfies $|S \setminus S_k| + |S_k \setminus S| \leq M$ cannot exceed $\min\{|S| + M, |V|\}$. Then, given an instance $\langle G, m\rangle$ of the maximum subgraph problem, construct an instance $\langle G_k^*, S^*, M^*, q^*\rangle$ of second-stage maximum subgraph problem with $G_k^* = G$, $S^* = \{i\}$ for a fixed $i \in V(G)$, $M^* = m-1$, and $q^* = m$. The order of the largest Π-subgraph S_k^* of the instance $\langle G_k^*, S^*, M^*, q^*\rangle$ never exceeds m due to the above observation, and is equal to m if and only if there exists a Π-subgraph of G of order m that contains vertex i. Thus, existence of a Π-subgraph of order m in G can be determined by solving no more than $|V(G)|$ instances $\langle G, \{i\}, m-1, m\rangle$ of the second-stage problem as described above. ∎

Note that while the introduced model assumes a common property Π for the subgraphs selected during both decision stages, possible extensions may include distinct properties at each stage. Further, the model may be enhanced by imposing non-uniform cost structures associated with selecting, adding and removing the vertices, or by introducing different budgetary restrictions in different scenarios.

3. A COMBINATORIAL BRANCH-AND-BOUND SOLUTION TECHNIQUE FOR THE TWO-STAGE STOCHASTIC MAXIMUM SUBGRAPH PROBLEM

In this section, we introduce an exact graph-based, or *combinatorial*, branch-and-bound (BnB) algorithm for solving the two-stage stochastic maximum subgraph problem (5)–(6). The proposed BnB technique relies on the nested representation (6), and, in view of the complexity analysis presented above, comprises both a first- and a second-stage BnB algorithm for problems (6a) and (6b), respectively. Traditional implementations of combinatorial BnB methods for maximum Π problems, such as the maximum clique problem, and so on [10, 17, 21, 27], maintain a *partial solution* set S, which contains vertices that induce a Π-subgraph, or a feasible solution to the original problem, and a *candidate set* C from which vertices are removed and added to S during branching. The bounding step involves analysis of the current feasible solution S and the candidate set C so as to obtain an upper bound on the size of the largest Π-subgraph that can be constructed by adding vertices from C to S.

In the context of the proposed BnB method for the two-stage stochastic maximum subgraph problem presented below, it is assumed that S_0 and S_k, $k \in \mathcal{N}$, represent first- and second-stage Π-subgraphs from which vertices are added and removed as the algorithms navigate their respective search space. Note that sets S_0, ..., S_N do not necessarily constitute a feasible solution to the TSMS problem (5)–(6), that is, they may not satisfy the budget constraints (5c) at intermediate stages of the algorithm. A collection of Π-subgraphs $S_0, \ldots, S_N$ becomes an incumbent solution if these subgraphs satisfy the budget constraints (5c) and the corresponding objective value exceeds the objective values of prior feasible solutions, thereby setting the new value of the global lower bound. The algorithms terminate after exhausting their respective search spaces, or BnB trees (note that large portions of these may be pruned, especially at later stages).

The first- and second-stage algorithms both work by navigating between *levels* of their BnB trees, where the level, or depth, of any tree node is indexed using a nonnegative integer ℓ that is defined by the cardinality of the current partial solution. We let the level of the root node be $\ell = 0$. An algorithm branches into a deeper level of the BnB tree whenever a *branching vertex* is added to the associated partial solution. Branching vertices are selected from a candidate set that contains vertices that can be added to the partial solution without violating the property Π. Similarly, an algorithm *backtracks* to a lower level whenever a previously added branching vertex is removed from the current partial solution.

The first-stage BnB algorithm begins by identifying a first-stage Π-subgraph S_0 in G_0 that satisfies property Π. It is then determined whether S_0 is reparable within the budget limit M in each of the second-stage scenarios $k \in \mathcal{N}$. If the latter condition holds, then a first-stage bounding condition determines if the objective value in (6a) can potentially be improved by solving the corresponding second-stage

problems. Whenever such potential improvement is possible, the second-stage BnB algorithm solves the second-stage problems $Q_k(S_0)$, $k \in \mathcal{N}$ (6b) by finding the largest possible Π-subgraphs S_k in G_k that can be obtained by removing or adding vertices to S_0 within the repair budget M. In this case, bounding conditions on the sizes of the second-stage subgraphs rely on the number of vertices that can be added and removed from S_0 without exceeding M. This search procedure repeats until the subgraphs S_0 and S_k, $k \in \mathcal{N}$, that maximize the objective of (5)–(6) are found.

A cornerstone element of the described above BnB procedure is an ability to construct an upper bound on the order of the largest Π-subgraph contained in a given graph:

Definition 3. *Given a subset $S \subseteq V$, let $\upsilon_\Pi(G[S])$ represent an upper bound on the order of the largest possible Π-subgraph contained in the induced graph $G[S]$:*

$$\upsilon_\Pi(G[S]) \geq |argmax_{S' \subseteq S} \left\{ |S'| : G[S'] \ni \Pi \right\}|.$$

The subscript Π in $\upsilon_\Pi(G[S])$ indicates that the properties and computation of this bound depend explicitly on Π.

Obviously, the best (i.e., tight) bound $\upsilon_\Pi(\cdot)$ can be obtained by solving the maximum subgraph problem. Therefore, in practice it is necessary to select a bounding method, that is, the method for computing $\upsilon_\Pi(\cdot)$, that achieves a necessary balance between the computational cost and the quality (tightness) of the obtained bound. Computational cost considerations necessarily require that $\upsilon_\Pi(\cdot)$ should be polynomially computable.

Finally, we would like to emphasize that the BnB methods for the TSMS problem that are presented next are rather general, and their computational efficiency will depend heavily on the particular hereditary property Π and the corresponding branching and bounding criteria used for processing of the search space. An illustration of the proposed procedure is furnished in section 4 for the case when Π represents the completeness property of a subgraph.

3.1. First-Stage Branch-and-Bound Algorithm

The first-stage BnB algorithm is initialized at level $\ell = 0$ with a partial solution $S_0 := \emptyset$, and partial and global lower bounds on the objective value of problem (6a), $\mathcal{Z} := -\infty$ and $\mathcal{Z}^* := -\infty$, respectively. It begins by branching to form a partial solution (subgraph) S_0 such that $G_0[S_0]$ satisfies property Π. By employing the upper bound $\upsilon_\Pi(\cdot)$ (see Definition 3), it is then *estimated* if the property Π of S_0 can be restored/repaired by *removing* no more than M vertices in the second-stage scenarios $k \in \mathcal{N}$. If S_0 is deemed reparable, a bounding criteria verifies whether an improvement in the objective value of problem (6a) is possible with respect to the maximum size that S_0 may become by adding more vertices to it (i.e., by branching further).

If an improvement in the objective value is not possible, or if S_0 is not reparable in some scenario $k \in \mathcal{N}$, the corresponding BnB node is fathomed and the algorithm backtracks by removing the last branching vertex that was added to S_0. If, however, the bounding criteria demonstrates that an improvement in the objective value is possible, the second-stage BnB described in section 3.2 solves problems $Q_k(S_0)$, $k \in \mathcal{N}$, to find the largest second-stage subgraphs $S_1, \ldots, S_k$. Assuming feasible second-stage subgraphs are indeed found, the resulting objective value of problem (6a) is stored if it is greater than any previously obtained objective value. The algorithm then branches to form a new partial solution (i.e., a BnB node) if there exist at least one vertex in the candidate set whose inclusion in S_0 would not violate property Π. If no such vertices exist in the candidate set, the algorithm backtracks.

Any time the algorithm backtracks, it subsequently branches to form a new partial solution if such a solution exists. If a new partial solution S_0 is obtained, the reparability check and bounding condition are evaluated at the corresponding node of the BnB tree. The described process repeats until the first-stage search space has been processed. The details of the first-stage BnB algorithm are presented next.

3.1.1. Reparability of S_0. Whenever a partial solution S_0 is constructed, it is first determined whether S_0 can be sufficiently modified in the second-stage scenarios so that the resulting subgraphs satisfy property Π. Observe that the most "favorable" realization of uncertainties is such that the structure of edge sets E_k, $k \in \mathcal{N}$, would preserve the property Π of S_0 in each $G_k[S_0]$. In such cases, no modifications (repairs) in the form of removing vertices from S_0 in the second-stage would be required.

Obviously, the modified sets of edges E_k, $k \in \mathcal{N}$, will generally not preserve the property Π of S_0 as described. Assuming otherwise would disregard structural variations between G_0 and G_k, particularly relative to how well solution S_0 will "perform" in any given scenario realization $k \in \mathcal{N}$. It is, therefore, of interest to introduce several feasibility and reparability conditions in the context posed by the following question: *given a current first-stage solution S_0, what is the minimum number of modifications that must be made to S_0 in any second-stage scenario $k \in \mathcal{N}$ in order to ascertain property Π?*

Prior to solving the second-stage problems $Q_k(S_0)$ for $k \in \mathcal{N}$, one possibility is to perform the feasibility test furnished by the next proposition.

Proposition 3 (Infeasibility certificate)**.** *For a given scenario $k \in \mathcal{N}$, let $S_0^{(k)}$ represent a subset of S_0 that induces a Π-subgraph in $G_k[S_0]$. If the following condition is satisfied,*

$$|S_0| - \max_{S_0^{(k)} \subseteq S_0} \left\{ |S_0^{(k)}| : G_k[S_0^{(k)}] \ni \Pi \right\} > M, \qquad (8)$$

then subgraph S_0 is an infeasible (irreparable) first-stage solution to problem (5)–(6).

Proof. Recall that the induced subgraph $G_0[S_0]$ has property Π by construction. Clearly, since the vertices remain

fixed between the decision stages, the largest possible set of vertices $S_0^{(k)}$ such that $G_k[S_0^{(k)}] \ni \Pi$ is no larger than $|S_0|$ (i.e., $S_0^{(k)} \subseteq S_0$). Hence, the left-hand side of expression (8) represents the smallest number Δ_k^- of vertices that must be removed from S_0 in order to obtain a subset $S_0^{(k)}$ that induces a subgraph $G_k[S_0^{(k)}]$ with property Π under scenario $k \in \mathcal{N}$. This immediately implies that if condition (8) holds for any $k \in \mathcal{N}$, the budget constraint in (6b) cannot be satisfied. ∎

If expression (8) is violated, then the expression in its left-hand side provides the minimum number of vertices that must be removed from S_0 in order to restore property Π in scenario $k \in \mathcal{N}$. However, finding the maximum subset $S_0^{(k)}$ in (8) by solving an $\mathcal{NP}$-complete problem of type (1) for each scenario $k \in \mathcal{N}$ is clearly computationally infeasible. Instead, we utilize the fact that $|S_0| \geq \upsilon_\Pi(G_k[S_0]) \geq |S_0^{(k)}|$, and employ a more tractable condition by replacing the second term in expression (8) by $\upsilon_\Pi(G_k[S_0])$:

Corollary 1 (Infeasibility certificate approximation)**.** *If the following condition is satisfied for a given scenario $k \in \mathcal{N}$,*

$$|S_0| - \upsilon_\Pi(G_k[S_0]) > M, \tag{9}$$

then (8) is also satisfied for the same k, and subgraph S_0 is an infeasible (irreparable) first-stage solution to problem (5)–(6).

If subgraph S_0 is deemed feasible under the approximate condition (9), the left-hand side of (9) represents an approximation of the minimum number of vertices that must be removed from S_0 under scenario $k \in \mathcal{N}$.

3.1.2. Candidate Set Generation and Refinement. At the current node of the BnB tree, level ℓ is associated with the candidate set $C_\ell \subseteq V$ from which any single vertex can be added to the partial solution S_0 without violating property Π. Branching is conducted by removing a branching vertex q from C_ℓ and adding it to S_0. The algorithm is initialized with $C_0 := V$, and once a branching vertex $q \in C_\ell$ is selected, the candidate set at level $\ell + 1$ is constructed by eliminating all the vertices from C_ℓ whose inclusion in S_0 would violate the property Π:

$$C_{\ell+1} := \{i \in C_\ell : G_0[S_0 \cup i] \ni \Pi\}. \tag{10}$$

Whenever the algorithm backtracks, the vertex q that was selected prior to constructing $C_{\ell+1}$ is removed from S_0.

Provided that $C_{\ell+1}$ is not an empty set, by virtue of Proposition 3, it is possible to determine the number of vertices that will have to be removed from subgraph S_0 in the second stage if a vertex $i \in C_{\ell+1}$ is added to S_0 in the first stage.

Proposition 4 (Candidate vertex infeasibility certifcate). *If inequality (8) or (9) is violated in scenario $k \in \mathcal{N}$, then vertex $i \in C_{\ell+1}$ can be removed from $C_{\ell+1}$ if the condition*

$$|S_0 \cup i| - \max_{S_i^{(k)} \subseteq S_0 \cup i} \left\{ |S_i^{(k)}| : G_k[S_i^{(k)}] \ni \Pi \right\} > M, \tag{11}$$

holds for some Π-subgraph $S_i^{(k)}$ in the induced subgraph $G_k[S_0 \cup i]$

Proof. The statement follows immediately from Proposition 3. ∎

As before, the above statement holds if the second term in (11) is replaced by an approximation:

Corollary 2. *The statement of Proposition 4 holds if inequality (11) is replaced by an approximate condition*

$$|S_0 \cup i| - \upsilon_\Pi(G_k[S_0 \cup i]) > M. \tag{12}$$

In addition to reducing the search space via removal of potential branching vertices at level $\ell + 1$, a refined candidate set can produce a more conservative upper bound on the objective value of problem (6a), as shown in the next subsection. Also, a brief discussion on the complexity associated with generating candidate sets is furnished in section 3.3. In what follows, we implicitly assume that the candidate set $C_{\ell+1}$ can be constructed in polynomial time.

3.1.3. Bounding If inequality (9) is violated for all scenarios $k \in \mathcal{N}$ at the current node of the BnB tree, then prior to solving the second-stage problems $Q_k(S_0)$, $k = 1, \ldots, N$, an upper bound on the objective value of problem (6a) is determined.

Proposition 5 (Myopic bounds). *Given a partial solution S_0 and a candidate set $C_{\ell+1}$, the expression*

$$\upsilon_\Pi(G_0[S_0 \cup C_{\ell+1}]) + \min\{\upsilon_\Pi(G_0[S_0 \cup C_{\ell+1}]) + M,\ |V|\} \tag{13}$$

provides an upper bound on the objective value for problem (6a) that can result from any Π subgraph in the set $S_0 \cup C_{\ell+1}$

Proof. By definition, the term $\upsilon_\Pi(G_0[S_0 \cup C_{\ell+1}])$ provides an upper bound on the size of the largest Π subgraph in $G_0[S_0 \cup C_{\ell+1}]$ by branching on the vertices in $C_{\ell+1}$. Hence, it is the maximum value that can be achieved in the first stage by branching on vertices in $C_{\ell+1}$.

Recall that a "most favorable" realization of uncertainties would preserve the property Π of the largest Π subgraph in $G_0[S_0 \cup C_{\ell+1}]$. Therefore, no modifications in the form of removing vertices in the second stage would be required, and the upper bound on the size of the largest Π subgraph in each graph $G_k[S_0 \cup C_{\ell+1}]$ would still be $\upsilon_\Pi(G_0[S_0 \cup C_{\ell+1}])$. For every scenario $k \in \mathcal{N}$, additionally assume that sufficiently many favorable edge modifications occur such that the budget M can exclusively be used to add new vertices to subgraph S_0. In other words, the second term of expression (13) represents an upper bound on the potential contribution of the recourse action under "ideal" circumstances. The statement of the proposition immediately follows. ∎

Expression (13) can be enhanced by taking into account the structural variations between G_0 and the second-stage graphs G_k, $k \in \mathcal{N}$.

Proposition 6 (Scenario-based bounds). *In the case when inequality (9) is violated for all scenarios $k \in \mathcal{N}$, the following expression provides an upper bound on the objective value of problem (6a):*

$$\upsilon_\Pi(G_0[S_0 \cup C_{\ell+1}]) + \sum_{k\in\mathcal{N}} p_k \min\{\upsilon_\Pi(G_k[S_0 \cup C_{\ell+1}]) + M_k, |V|\}, \quad (14)$$

where the $M_k = M - (|S_0| - \upsilon_\Pi(G_k[S_0]))$, $k \in \mathcal{N}$, represent reduced budgets obtained from (9).

Proof. The first term in the left-hand side of (14) follows the same logic as in Proposition 5. In the second term, $\upsilon_\Pi(G_k[S_0 \cup C_{\ell+1}])$ represents the upper bounds on the largest Π subgraph in $G_k[S_0 \cup C_{\ell+1}]$. Notice that if inequality (9) is violated for all scenarios $k \in \mathcal{N}$, then $(|S_0| - \upsilon_\Pi(G_k[S_0]))$ gives the minimum number of vertices that must be removed from S_0 in scenario k in order to preserve property Π in the second stage. A reduced budget $M_k = M - (|S_0| - \upsilon_\Pi(G_k[S_0]))$ represents the maximum number of vertices that can be added in scenario k. Hence, one immediately obtains

$$Q_k(S_0) \leq \min\{\upsilon_\Pi(G_k[S_0 \cup C_{\ell+1}]) + M_k, |V|\},$$

and expression (14) readily follows by considering all scenarios $k \in \mathcal{N}$. ∎

The last proposition readily implies that if inequality

$$\upsilon_\Pi(G_0[S_0 \cup C_{\ell+1}]) + \sum_{k\in\mathcal{N}} p_k \min\{\upsilon_\Pi(G_k[S_0 \cup C_{\ell+1}]) + M_k, |V|\} \leq \mathcal{Z}^* \quad (15)$$

is violated, the algorithm proceeds to solve the second-stage recourse problems (6b) for all $k \in \mathcal{N}$, otherwise the algorithm backtracks from the current node of the BnB tree.

In cases when inequality (15) is indeed violated, then there are two possibilities that can arise with respect to the second-stage problems (6b). First, the problem (6b) may be infeasible for some k given the current solution S_0. Then, the corresponding second-stage function $Q_k(S_0)$ and the respective recourse function $E_\omega[Q(S_0)] = \sum_{k\in\mathcal{N}} p_k Q_k(S_0)$ assume the value of $-\infty$. In this case, the algorithm backtracks by removing the most recent branching vertex q, and the next branching vertex is selected from the candidate set if $C_\ell \neq \emptyset$. An illustration of such a case is given in Figure 1. Alternatively, all second-stage problems are feasible and the functions $Q_k(S_0)$, $k = 1, \ldots, N$, are finite, whence the current objective value associated with problem (6a) is updated as $\mathcal{Z} = |S_0| + \sum_{k\in\mathcal{N}} p_k Q_k(S_0)$; the global lower bound $\mathcal{Z}^*$ is replaced by $\mathcal{Z}$ if $\mathcal{Z}^* < \mathcal{Z}$. Then, if the candidate set is nonempty, $C_{\ell+1} \neq \emptyset$, the algorithm selects a branching vertex q from the next level $\ell + 1$. The branching vertex q at level ℓ is stored as q_ℓ for backtracking purposes. Alternatively, if $C_{\ell+1} = \emptyset$, the algorithm backtracks by removing vertex q from S_0.

Whenever condition (15) is satisfied, there is no possibility of achieving an improvement over the global lower bound $\mathcal{Z}^*$ by exploring further levels of the BnB tree; vertex q is removed from S_0. If $C_\ell = \emptyset$, the algorithm backtracks to level $\ell - 1$ by removing from S_0 the most recent branching vertex that was used at level $\ell - 1$, namely vertex $q_{\ell-1}$. The described first-stage BnB procedure is formalized in Algorithm 1.

3.2. *Second-Stage Branch-and-Bound Algorithm*

If condition (15) is violated, a second-stage BnB procedure is used to solve problem (6b) for each scenario. Namely, the algorithm solves the second-stage problem $Q_k(S_0)$, $k \in \mathcal{N}$, by identifying the largest subgraph $S_k \subseteq V(G_k)$ with property Π that satisfies the budgetary limit imposed by constraint (5c). The optimal solution S_k^* must be feasible with respect to the first-stage partial solution S_0 in the sense that the total number of vertices added to and removed from S_0 does not exceed the budget M. The bounding procedure pertains to eliminating unfavorable search space relative to the budgetary limit M and the property Π.

As in the first-stage BnB scheme, the second-stage algorithm moves between levels of the (second-stage) BnB tree by exploring branching vertices from candidate sets that individually satisfy the property Π with respect to the partial solution S_k. It begins by selecting a branching vertex q from the candidate set C_ℓ^k, which is initially $C_0^k := V$.

Due to the fact that discrepancies between S_0 and S_k impose a budgetary penalty, the natural tendency is to maintain as similar a structure as possible in the second stage. Noting that vertices common to C_ℓ^k and the solution S_0 do not utilize the budget M, a branching vertex $q \in \{S_0 \cap C_\ell^k\}$ is always selected first if $\{S_0 \cap C_\ell^k\} \neq \emptyset$. Once q is added to the second-stage partial solution S_k, the candidate set at the next level $C_{\ell+1}^k$ is constructed by removing all the vertices from C_ℓ^k whose inclusion in S_k would violate property Π.

3.2.1. Budgetary Bounding. Given the first-stage and second-stage partial solutions S_0 and S_k, respectively, the cost of S_k according to constraint (5c) can easily be computed so that $\delta = |S_0 \setminus S_k| + |S_k \setminus S_0|$. Observe that the number of branching vertices in $C_{\ell+1}^k$ that could reduce the value of δ at consecutive levels of the BnB tree is given by $|S_0 \cap C_{\ell+1}^k|$. Therefore, due to the fact that the largest subgraph in $G_k[S_0 \cap C_{\ell+1}^k]$ that satisfies the property Π is bounded by $\upsilon_\Pi(G_k[S_0 \cap C_{\ell+1}^k])$, the maximum number of vertices that

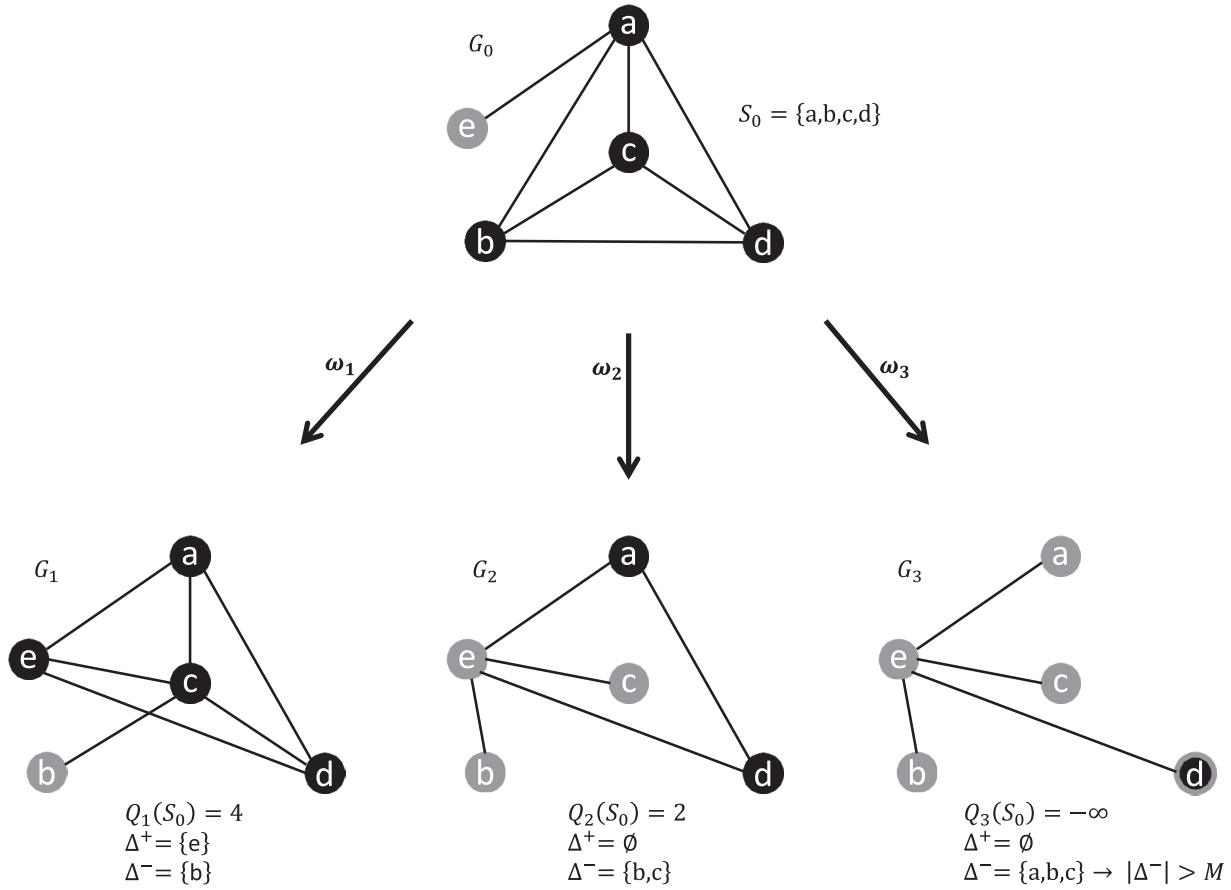

FIG. 1. An example with three scenarios demonstrating the reparability of subgraph S_0 with a repair budget $M = 2$ and property Π representing completeness. Black vertices represent those belonging to a complete subgraph. Observe that solution S_0 is feasible (reparable) with respect to scenarios ω_1 and ω_2, but is infeasible (not reparable) with respect to scenario ω_3. Scenario ω_2 also illustrates that the subgraphs in the first or second stages need not be maximal.

could potentially be used to reduce the cost δ is likewise given by $\gamma = v_\Pi(G_k[S_0 \cap C^k_{\ell+1}])$. Several budgetary considerations emerge as a result.

The following conditions are possible when $\delta - \gamma \leq M$:

C1. If $\delta \leq M$, then (5c) is satisfied via vertices in S_k, and S_k replaces S^*_k if $|S_k| > |S^*_k|$. In cases when $\delta = M$ and $\gamma > 0$, a branching vertex $q \in \left\{S_0 \cap C^k_{\ell+1}\right\}$ is selected and the algorithm branches to level $\ell := \ell + 1$. Conversely, if $\gamma = 0$, adding more vertices to S_k will violate (5c); thus, the algorithm backtracks by removing the most recent branching vertex q from S_k. If $\delta < M$ and $C^k_{\ell+1} \neq \emptyset$, the algorithm always branches.

C2. If $\delta > M$, the partial solution S_k is infeasible with respect to (5c). However, the set $\left\{S_0 \cap C^k_{\ell+1}\right\}$ necessarily contains a sufficient number of vertices to (potentially) satisfy M at deeper levels of the BnB tree, that is, $\gamma \geq \delta - M$. The algorithm branches accordingly.

In cases when $\delta - \gamma > M$, restriction (5c) cannot be satisfied by exploring the vertices in $C^k_{\ell+1}$, and, therefore, the algorithm backtracks as before.

If the above budgetary condition is indeed satisfied relative to the partial solution and candidate set (i.e., if $\delta - \gamma \leq M$), an additional upper bound on the size of the partial solution is applied. Notice that if γ vertices can potentially be added to the partial solution from the set $\left\{S_0 \cap C^k_{\ell+1}\right\}$ at deeper levels of the BnB tree, then at most $M - (\delta - \gamma)$ vertices can be added from the set $\left\{C^k_{\ell+1} \setminus S_0\right\}$ without violating the budget M. Therefore, we check the following condition at the current node of the BnB tree:

$$|S_k| + \min\left\{\gamma + M - (\delta - \gamma),\ v_\Pi(G_k[C^k_{\ell+1}])\right\} > |S^*_k|.$$

The function "min" selects whichever term provides the lowest upper bounds on the candidate set, where the first term restricts the bounds via the budget limit, while the second considers the largest possible subgraph of property Π contained in the candidate set.

Algorithm 2 outlines the described solution technique for the second-stage problem $Q_k(S_0)$, $k \in \mathcal{N}$.

3.3. Optimality and Further Complexity Discussion. The fact that the described above two-stage combinatorial BnB algorithm terminates with an optimal solution can be seen from the nested formulation (6a), where the TSMS

Algorithm 1 First-stage combinatorial BnB method

Initialize: $\ell := 0$; $C_0 := V$; $S_0 := \emptyset$;
  $\mathcal{Z} = \mathcal{Z}^* = -\infty$; $M \in \mathbb{Z}_+$;
while $\ell \geq 0$ **do**
  if $C_\ell \neq \emptyset$ **then**
    select a vertex $q \in C_\ell$;
    $C_\ell := C_\ell \setminus q$;
    $S_0 := S_0 \cup q$;
    for $k \in \mathcal{N}$ **do**
      if $|S_0| - \upsilon_\Pi(G_k[S_0]) > M$ **then**
        $S_0 := S_0 \setminus q$;
        goto Step 3
      else
        $M_k := M - (|S_0| - \upsilon_\Pi(G_k[S_0]))$
    $C_{\ell+1} := \{i \in C_\ell : G_0[S_0 \cup i] \ni \Pi\}$;
    for $i \in C_{\ell+1}$ **do**
      if $\left\{k \in \mathcal{N} : |S_0 \cup i| - \upsilon_\Pi(G_k[S_0 \cup i]) > M\right\} \neq \emptyset$ **then**
        $C_{\ell+1} := C_{\ell+1} \setminus i$
    if $\upsilon_\Pi(G_0[S_0 \cup C_{\ell+1}]) + \sum_{k\in\mathcal{N}} p_k \min\left\{\upsilon_\Pi(G_k[S_0 \cup C_{\ell+1}]) + M_k, |V|\right\} > \mathcal{Z}^*$ **then**
      for $k \in \mathcal{N}$ **do**
        compute $Q_k(S_0)$;
        if $Q_k(S_0) = -\infty$ **then**
          $S_0 := S_0 \setminus q$;
          goto Step 3;
        else
          $\mathcal{Z} := |S_0| + \sum_{k\in\mathcal{N}} p_k Q_k(S_0)$;
      if $\mathcal{Z} > \mathcal{Z}^*$ **then**
        $\mathcal{Z}^* := \mathcal{Z}$;
      if $C_{\ell+1} \neq \emptyset$ **then**
        $q_\ell := q$
        $\ell := \ell + 1$;
      else
        $S_0 := S_0 \setminus q$;
    else
      $S_0 := S_0 \setminus q$
  else
    $S_0 := S_0 \setminus q_{\ell-1}$;
    $\ell := \ell - 1$;
return $\mathcal{Z}^*$;

Algorithm 2 Second-stage combinatorial BnB method for computing $Q_k(S_0)$

Input: G_k; S_0;
Initialize: $\ell := 0$; $C_0^k := V$; $S_k := \emptyset$; $S_k^* := \emptyset$;
while $\ell \geq 0$ **do**
  if $C_\ell^k \neq \emptyset$ **then**
    if $|S_0 \cap C_\ell^k| \neq \emptyset$ **then**
      select a vertex $q \in \{S_0 \cap C_\ell^k\}$;
    else
      select a vertex $q \in C_\ell^k$;
    $C_\ell^k := C_\ell^k \setminus q$;
    $S_k := S_k \cup q$;
    $C_{\ell+1}^k := \{i \in C_\ell^k : G_k[i \cup S_k]$ satisfies $\Pi\}$;
    $\delta := |S_0 \setminus S_k| + |S_k \setminus S_0|$;
    $\gamma := \nu_\Pi\big(G_k[S_0 \cap C_{\ell+1}^k]\big)$;
    if $\delta - \gamma \leq M$ **and** $|S_k| + \min\left\{\gamma + M - (\delta - \gamma),\ \nu_\Pi\big(G_k[C_{\ell+1}^k]\big)\right\} > |S_k^*|$ **then**
      if $\delta = M$ **and** $\gamma = 0$ **then**
        if $|S_k| > |S_k^*|$ **then**
          $S_k^* := S_k$;
        $S_k := S_k \setminus q$;
      else
        $q_\ell := q$;
        $\ell := \ell + 1$;
        if $\delta \leq M$ **and** $|S_k| > |S_k^*|$ **then**
          $S_k^* := S_k$;
    else
      $S_k := S_k \setminus q$;
  else
    $S_k := S_k \setminus q_{\ell-1}$;
    $\ell := \ell - 1$;
return $Q_k(S_0) := |S_k^*|$;

problem has the form of a deterministic maximum Π problem with a complex nonlinear objective. First, note that the TSMS problem (5)–(6) is always feasible if the property Π is nontrivial, that is, it is satisfied by a single-vertex graph (see Definition 2). Then, observe that once the processing of the first-stage solution space is complete, the obtained partial solution $S_0 \neq \emptyset$ will represent a Π-subgraph that has the highest objective value in the sense of (6a) among all processed partial solutions (i.e., nodes of the first-stage BnB tree). The nodes of the BnB tree that have not been processed (i.e., were fathomed) correspond to partial solutions that either failed the reparability check, or whose objective upper bound did not exceed the current lower bound.

Finally, we would like to comment on the complexity of candidate set generation when the sought property Π is hereditary. As it could be readily seen, the operation of constructing candidate set $C_{\ell+1}$ from the preceding candidate set C_ℓ constitutes one of the basic steps of the first- and second-stage BnB algorithms. Consequently, the computational cost of this step can significantly affect the computational performance of the solution method. In this regard, a major question is whether one can efficiently verify property Π for any given subgraph. The associated decision problem is as follows: given a subgraph S, determine whether S satisfies property Π, or whether some fraction of the representation of S can

be modified in order for S to satisfy property Π. In the latter case, it is said that S is ε- *far* from satisfying property Π, where ε corresponds to the fraction of modifications that need to be made. With respect to hereditary properties, a substantial body of literature has accumulated in recent years to address this question. For example, Alon and Shapira [2] showed that every hereditary property is testable with one-sided error. Further, several characterizations of hereditary properties have been proposed in [14]. As described above, a property Π is said to be *node-hereditary* if it is closed under taking *induced* subgraphs of G, and is *subgraph-hereditary* if it is closed under taking subgraphs of G. A property is *minor-hereditary* if any *graph minor*[1] S of graph G satisfies Π. In a series of seminal studies [24, 25], Robertson and Seymour established the *graph minor theorem* which, among others, predicated polynomial time identification of hereditary properties closed under graph minors.

4. COMPUTATIONAL STUDY: A TWO-STAGE STOCHASTIC MAXIMUM CLIQUE PROBLEM

As an illustrative example of the general TSMS problem and the proposed solution approaches, in this section we consider the *two-stage stochastic maximum clique problem*, a special case of the TSMS problem (5) when the property Π represents completeness. The corresponding mathematical programming formulation that we use in this work employs the well-known *edge formulation* [22] of the structural constraints that guarantee completeness of the selected subgraph, namely

$$\begin{aligned}&\left\{\mathbf{z}\in\{0,1\}^{|V|}:\mathbf{\Pi}_G(\mathbf{z})\leq\mathbf{0}\right\}\\&\quad=\left\{\mathbf{z}\in\{0,1\}^{|V|}:z_i+z_j\leq 1 \text{ for all } (i,j)\in\bar{E}\right\},\end{aligned}$$

where $\bar{E}$ represents the set of edges of the complement of graph G, that is, $(i,j)\in E \Leftrightarrow (i,j)\notin\bar{E}$ for any $i,j\in V$. Then, the two-stage stochastic maximum clique problem admits the following 0-1 integer programming form:

$$\max \quad \sum_{i\in V} x_i + \sum_{k\in\mathcal{N}} p_k\Big(\sum_{i\in V} y_{ik}\Big) \tag{16a}$$

$$\text{s.t.} \quad x_i+x_j\leq 1, \qquad \forall (i,j)\in\bar{E} \tag{16b}$$

$$y_{ik}+y_{jk}\leq 1, \qquad \forall (i,j)\in\bar{E}_k,\ k\in\mathcal{N} \tag{16c}$$

$$\sum_{i\in V}|x_i-y_{ik}|\leq M, \quad \forall k\in\mathcal{N} \tag{16d}$$

$$x_i,y_{ik}\in\{0,1\}, \qquad \forall i\in V,\ k\in\mathcal{N}, \tag{16e}$$

where the first-stage binary decision variables are defined as $x_i=1$ if the vertex $i\in V$ belongs to S_0, and $x_i=0$ otherwise. Similarly, for scenario $k\in\mathcal{N}$, the second-stage decision variables are defined as $y_{ik}=1$ if $i\in V$ belongs to S_k, and $y_{ik}=0$ otherwise. Formulation (16) can be solved with appropriate integer programming solvers.

The property-specific techniques for finding cliques in all types of graphs via the BnB procedure in section 3 and Algorithms 1–2 are described next.

4.1. Candidate Set Generation, Branching, and Bounding Techniques

When property Π defines a clique, a number of efficient techniques have been developed in the literature that can be utilized for candidate set generation, branching, and bounding. For example, the candidate sets can efficiently be generated and updated via an intersection of neighboring vertices common to the clique elements. Constructing candidate set (10) is performed by pairwise testing any vertex $j\in S_0$ against a vertex $i\in C_\ell$, and removing the vertices from C_ℓ that are not adjacent to subgraph S_0, that is,

$$C_{\ell+1}:=\{i\in C_\ell:(i,j)\in E_0,\forall j\in S_0\}.$$

A refinement criterion with respect to the second-stage graph scenarios as described by Corollary 4 is furnished by the next proposition.

Proposition 7. *Given a scenario $k\in\mathcal{N}$ and a vertex $i\in C_{\ell+1}$, let $\Gamma_k(i):=\{j\in S_0:(i,j)\in E_k\}$ represent the (sub)set of vertices such that any two vertices i, j are adjacent in $G_k[S_0\cup i]$. If the following inequality holds,*

$$|S_0|-|\Gamma_k(i)|>M, \tag{17}$$

then vertex i can be removed from $C_{\ell+1}$

Proof. If i is added to S_0 in the first stage, then it is easy to see that the vertices $S_0\setminus\Gamma_k(i)$ must be removed from S_0 in order for $G_k[\Gamma_k(i)\cup i]$ to (possibly) form a complete graph in scenario $k\in\mathcal{N}$. Note that if subgraph $G_k[\Gamma_k(i)]$ is not a clique in $k\in\mathcal{N}$, then at least one vertex from the set $S_0\setminus\Gamma_k(i)$ must be further removed from S_0 in the first stage. Thus, $|\Gamma_k(i)\cup i|$ provides an upper bound on the size of the maximum clique contained in $G_k[S_0\cup i]$. Consequently, expression (17) approximates the minimum number of vertices that must be removed from S_0 in the second stage if vertex i is included, which cannot exceed the budget M. ∎

In this study, we consider two techniques for computing the upper bound $\upsilon_\Pi(\cdot)$ on the size of a maximum clique and for selecting a branching vertex $q\in C_\ell$ when property Π represents a clique. We emphasize that proper selection of branching and bounding mechanisms according to a graph's structural characteristics and the sought property Π does heavily influence the computational performance of the solution method described in Algorithm 1.

4.1.1. An Approximate Coloring Algorithm. The first technique utilizes principles introduced by Tomita and Seki [27] to estimate the size of a maximum clique contained in

[1] A graph S is a minor of G if edge contractions can be performed on a subgraph of G to obtain S.

$G[S]$, $S \subseteq V$, by partitioning S into independent sets, also know as *numbering* or *coloring* classes. The vertices in S are first sorted in degree descending order, and a *minimum* positive integer n_i is assigned to each vertex $i \in S$ such that $n_i \neq n_j$ if the pair $i,j \in S$ are connected by an edge $(i,j) \in E(G)$. Consequently, vertices associated with a number class n_k (i.e., vertices with the same assigned integer value) form an independent set.

Since that the size of any clique embodied in $G[S]$ cannot exceed the number of coloring classes generated from S, one immediately obtains a bound on the maximum clique size as

$$\upsilon_\Pi(G[S]) = \max\{n_i : i \in S\}.$$

We use this expression in Algorithm 1 to obtain the bounds $\upsilon_\Pi(G_k[S_0])$ and $\upsilon_\Pi(G_k[S_0 \cup C_{\ell+1}])$, $k \in \{0\} \cup \mathcal{N}$. Condition (15) then takes the form:

$$|S_0| + \max\{n_i : i \in C_{\ell+1}\} + \sum_{k \in \mathcal{N}} p_k \min\left\{\max\{n_i : i \in S_0 \cup C_{\ell+1}\}_k + M_k, |V|\right\} \leq \mathcal{Z}^*. \tag{18}$$

The branching rule used in connection with the described approximate coloring scheme is as follows: select a vertex $q \in C_\ell$ with the maximum number $n_q := \max\{n_i : i \in C_\ell\}$. Note that an initial coloring of set $C_0 := V$ is performed prior to Step 2.

4.1.2. Directed Acyclic Path Decomposition. Yamaguchi and Masuda [33] proposed a clever technique for finding maximum *weighted* cliques in graphs by transforming $G[S]$, $S \subseteq V$, into a directed acyclic graph $\overrightarrow{G}[S]$ such that the lengths of the resulting acyclic paths represent bounds on the size of the maximum clique in $G[S]$. The method proceeds as follows. Without loss of generality, let each vertex $i \in S$ be associated with a unit weight $w_i = 1$, and define set $U(S) := \{u_i : i \in S\}$, where each element u_i is initially equivalent to w_i. Then, the set $U(S)$ is updated by sequentially "*propagating*" the elements u_i, $\forall i \in S$, onto adjacent members in S. Particularly, during each iteration a vertex i that corresponds to the minimum argument u_i in the set $U(S)$ is selected, and u_i is propagated by adding it to the weights of vertices $j \in S$ adjacent to vertex i in graph $G[S]$. The elements adjacent to u_j are updated as

$$u_j = \begin{cases} u_i + w_j, & \text{if } u_j < u_i + w_j, \\ u_j, & \text{otherwise,} \end{cases} \quad \text{for all } j \in \{j : (i,j) \in E,\ i,j \in S\}. \tag{19}$$

Once a vertex $i \in S$ has been processed, u_i is fixed and cannot be increased in subsequent propagations from other (unprocessed) adjacent vertices in S. The updating process terminates once all the elements in $U(S)$ have been fixed.

Observe that sequentially fixing elements u_i produces a directed acyclic graph $\overrightarrow{G}[S]$, where, once all the elements in $U(S)$ are fixed, any $u_i \in U(S)$ represents the longest acyclic path in $\overrightarrow{G}[S]$ whose endpoint is the vertex $i \in S$ (see [33] for details). Utilizing the fact that the length[2] of a longest path in $\overrightarrow{G}[S]$ is an upper bound on the maximum clique size in $G[S]$, one obtains the bounding condition $\upsilon_\Pi(G[S]) = \max\{u_i \in U(S)\}$. Expression (15) then takes the form:

$$|S_0| + \max\{u_i \in U(C_{\ell+1})\} + \sum_{k \in \mathcal{N}} p_k \min\left\{\max\{u_i \in U(S_0 \cup C_{\ell+1})\}_k + M_k, |V|\right\} \leq \mathcal{Z}^*. \tag{20}$$

In this case, it is assumed that a vertex with the largest propagated weight from adjacent vertices has a high probability of being a part of the maximum clique. As a result, the algorithm branches by selecting a vertex $q \in C_\ell$ that corresponds to the maximum element in $U(C_\ell)$.

4.2. Numerical Experiments and Results

Numerical experiments demonstrating the performance of the proposed BnB algorithms for solving the TSMS problem when property Π represents a clique were conducted. Problem (16) was solved for randomly generated Erdös-Rényi graphs of orders $|V| = 25, 50, 75, 100$ with average densities of $d = 0.2, 0.5, 0.8$. Similar experiments were also conducted on various DIMACS graph instances. For any given graph configuration, the number of vertices $|V|$ and densities d remained fixed during both decision stages. The value of the constant M in the budget constraints was determined by $M = \lceil \varepsilon \cdot \mathbb{E}[\omega(G)] \rceil$, where the term $\mathbb{E}[\omega(G)]$ represents the expected size of the maximum clique in a uniform random graph G with edge probability d [12]:

$$\mathbb{E}[\omega(G)] = \frac{2}{\ln(1/d)} \ln|V(G)| + o(\ln|V(G)|).$$

Unless specified otherwise, the constant ε was fixed at 0.5. The number of second-stage graph scenarios was selected as $N = 25$, 50, 75. In order to preserve the (expected) density, the modified graph scenarios were generated by allowing the existing edges in E_0 to fail with a probability of 0.5 and allowing $|E_0|$ randomly selected complement edges to form with the same probability.

The combinatorial first- and second-stage BnB algorithms described in section 3 were coded using C++, and CPLEX 12.5 integer programming (IP) solver was used for solving the mathematical programming formulation (16) of the two-stage stochastic maximum clique problem. The computations were run on an Intel Xeon 3.30GHz PC with 128GB of RAM, and version 12.5 of CPLEX solver in Windows 7 64-bit environment was used.

The combinatorial BnB method defined by Algorithms 1 and 2 was implemented in two versions, which use the branching and bounding techniques described in sections

[2] The path length is given by the aggregate weight of vertices that it coincides with.

TABLE 1. Average solution times (in seconds) and objective values for problem (16) on random graphs with an edge density of 0.2. Entries representing the best average running times are highlighted in bold

$d = 0.2$		CPLEX			BnB 4.1.1			BnB 4.1.2		
$\|V\|$	N	#	Time (s)	Objective	#	Time (s)	Objective (%)	#	Time (s)	Objective (%)
25	25	5	6.80	5.06	5	0.02	0	5	**0.00**	0
	50	5	22.91	4.95	5	0.04	0	5	**0.01**	0
	75	5	52.95	4.98	5	0.04	0	5	**0.01**	0
50	25	5	180.50	6.77	5	0.20	0	5	**0.04**	0
	50	5	951.08	6.56	5	0.37	0	5	**0.08**	0
	75	4	2218.67	6.57	5	0.60	0	5	**0.13**	0
75	25	4	2383.23	6.84	5	0.99	0	5	**0.20**	0
	50	0	—	6.51	5	2.53	3.4	5	**0.47**	3.4
	75	0	—	2.47	5	3.20	170.5	5	**0.74**	170.5
100	25	0	—	6.68	5	5.29	5.5	5	**0.71**	5.5
	50	0	—	1.94	5	12.05	251.9	5	**1.88**	251.9
	75	0	—	0.00	5	16.13	∞	5	**2.77**	∞

TABLE 2. Average solution times (in seconds) and objective values for problem (16) on random graphs with an edge density of 0.5. Entries representing the best average running times are highlighted in bold

$d = 0.5$		CPLEX			BnB 4.1.1			BnB 4.1.2		
$\|V\|$	N	#	Time (s)	Objective	#	Time (s)	Objective (%)	#	Time (s)	Objective (%)
25	25	5	13.49	10.58	5	0.73	0	5	**0.06**	0
	50	5	822.26	10.55	5	1.14	0	5	**0.11**	0
	75	2	804.32	10.29	5	2.41	0	5	**0.17**	0
50	25	4	1975.50	12.40	5	27.21	0	5	**2.89**	0
	50	3	3268.51	12.72	5	43.20	0.3	5	**5.08**	0.3
	75	0	—	12.21	5	50.96	3.0	5	**6.54**	3.0
75	25	0	—	13.26	5	312.48	6.0	5	**38.32**	6.0
	50	0	—	13.11	5	635.89	6.6	5	**87.22**	6.6
	75	0	—	13.08	5	835.85	6.9	5	**98.21**	6.9
100	25	0	—	13.55	5	1499.11	8.6	5	**205.96**	8.6
	50	0	—	13.25	3	2840.66	10.5	5	**369.51**	10.5
	75	0	—	6.70	0	—	118.5	0	**673.56**	118.6

4.1.1 and 4.1.2, and which are henceforth referred to as "BnB 4.1.1" and "BnB 4.1.2," respectively. The computational performance of both variants of Algorithm 1–2 was compared with that of the mathematical programming formulation (16) as solved by the CPLEX solver. The results are reported in Tables 1–5, where columns with headings "CPLEX," "BnB 4.1.1," and "BnB 4.1.2" contain the results obtained using the respective methods. Five instances of each problem/graph configuration were generated and the corresponding solution times (in seconds) and objective values were averaged accordingly. The entries representing the best average running times are highlighted in bold. For experiments involving Erdös-Rényi graphs, the average objective values are only given for CPLEX, whereas the percentage change in the average objective values with respect to CPLEX are reported for the BnB algorithms. A maximum solution time limit of 3,600 seconds was imposed and the symbol "—" is used to indicate that the time limit was exceeded for all ten instances for the given graph configuration. Columns corresponding to the symbol "#" provide the number of instances solved within the time limit. If only a portion of the instances were solved within the time limit, the number of instances that achieved a solution and their corresponding average solution times are presented.

Table 1 summarizes the computational times for graphs with average edge densities of $d = 0.2$. Observe that both BnB algorithms provide improvement in running time of at least two to three orders of magnitude on all problem configurations in comparison to the CPLEX IP solver, and the BnB variant based on acyclic path decomposition produces the best results. It must be noted, however, that sparse graphs put the mathematical programming formulation (16) of the two-stage stochastic maximum clique problem at a disadvantage, since the employed "edge formulation" of clique constraints is based on the complement of the graph, which results in a large number of constraints (16b)–(16c) when the underlying

TABLE 3. Average solution times (in seconds) and objective values for problem (16) on random graphs with an edge density of 0.8. Entries representing the best average running times are highlighted in bold

$d = 0.8$		CPLEX			BnB 4.1.1			BnB 4.1.2		
$\|V\|$	N	#	Time (s)	Objective	#	Time (s)	Objective (%)	#	Time (s)	Objective (%)
25	25	5	**1.29**	20.74	5	246.72	0	5	22.28	0
	50	5	**2.18**	21.24	5	650.78	0	5	56.08	0
	75	5	**4.23**	20.43	5	1197.33	0	5	61.70	0
50	25	0	—	28.94	0	—	−5.3	0	—	−1.3
	50	0	—	29.06	0	—	−7.7	0	—	−1.9
	75	0	—	28.66	0	—	−8.2	0	—	−2.8
75	25	0	—	33.80	0	—	−8.5	0	—	−3.3
	50	0	—	33.59	0	—	−6.6	0	—	−2.3
	75	0	—	32.18	0	—	−6.2	0	—	0.1
100	25	0	—	34.88	0	—	−3.8	0	—	−2.5
	50	0	—	33.53	0	—	0.9	0	—	4.1
	75	0	—	33.55	0	—	−0.2	0	—	5.1

TABLE 4. Average solution times (in seconds) and objective values for problem (16) on random graphs with 50 vertices, 25 scenarios, edge density of 0.2, and ε in the range $[0.1, 0.9]$. Entries representing the best average running times are highlighted in bold

$d = 0.2$			CPLEX			BnB 4.1.1			BnB 4.1.2		
$\|V\|$	N	ε	#	Time (s)	Objective	#	Time (s)	Objective (%)	#	Time (s)	Objective (%)
50	25	0.1	5	261.33	7.49	5	22.58	0	5	**0.70**	0
		0.3	5	272.74	10.76	5	68.44	0	5	**1.54**	0
		0.5	4	1975.50	12.40	5	27.21	0	5	**2.89**	0
		0.7	0	—	13.50	5	105.76	0.18	5	**7.68**	0.18
		0.9	0	—	14.30	5	371.30	0.34	5	**22.77**	0.34

TABLE 5. Solution times (in seconds) and objective values for problem (16) on DIMACS graphs. Entries representing the best average running times are highlighted in bold

DIMACS			CPLEX		BnB 4.1.1		BnB 4.1.2	
File (.clq)	$\|V\|$	d	Time	Objective	Time	Objective	Time (s)	Objective
Adjnoun	112	0.068	3474.13	6.52	42.18	6.52	**0.16**	6.52
Celegans Metabolic	453	0.020	—	∞	426.76	7.02	**17.96**	7.02
Celegans Neural	297	0.049	—	∞	—	7.08	**63.20**	7.14
Chesapeake	39	0.229	13.78	7.88	1.53	7.88	**0.03**	7.88
Dolphins	62	0.084	112.56	6.42	9.06	6.42	**0.03**	6.42
Football	115	0.094	—	7.68	171.80	8.28	**0.95**	8.28
Jazz	198	0.141	—	∞	—	9.5	**223.98**	9.54
Karate	34	0.139	10.87	6.48	2.82	6.48	**0.02**	6.48
Lesmis	77	0.087	1217.95	7.08	173.92	7.08	**0.61**	7.08
Polbooks	105	0.081	—	6.54	46.54	6.82	**0.25**	6.82

graph is sparse. At the same time, the proposed combinatorial BnB algorithm performs better when the "depth," or the number of levels of the BnB tree is smaller, which is observed on sparse graphs.

Thus, a more fair comparison of the combinatorial and mathematical programming-based schemes can be accomplished when one considers graphs with densities close to $d = 0.5$; see Table 2. It still can be observed, though, that the combinatorial BnB methods drastically outperform the mathematical programming formulation, and the branching and bounding rules based on acyclic path decomposition are again superior compared to vertex coloring. Nevertheless, graphs of density $d = 0.5$ present a greater challenge to the proposed BnB method, as the coloring-based branching and bounding variant was unable to solve all instances of the two largest graph configurations within the allowed time limit, even though the obtained average objective values were relatively close to the optimal solution values.

Computational results for the two-stage stochastic maximum clique problem on graphs with average densities of

$d = 0.8$ are presented in Table 3. At these densities, the combinatorial BnB methods are generally inferior to the mathematical programming formulation (16), which can be explained by the fact that the number of clique constraints (16b)–(16c) is relatively small for dense graphs, making problem (16) easier to solve. In contrast, the depth of the BnB tree increases with the density of the graph, which leads to deteriorated BnB solution times. However, the acyclic decomposition BnB algorithm generated noticeably better average objective values in comparison to CPLEX when $|V| = 100$, $N = 50, 75$, an obvious deviation from the trend of results associated with other graph configurations of the same density.

Table 4 furnishes computational results demonstrating the effects of changes in the budget $M = \lceil \varepsilon |V| \rceil$ for values of ε in the range [0.1, 0.9] when $|V| = 50$, $N = 25$ and $d = 0.2$. Irrespective of the solution technique, it is evident that the computation time generally increases as ε increases. This trend is consistent with the fact that the bounding condition (15) becomes weaker for larger budgets M due to the fact that a larger number of first-stage solutions may be repaired in the second stage. Consequently, a larger number of feasible second-stage solutions must be explored, thus increasing the size of the corresponding search space.

Finally, Table 5 presents the objective values and computation times obtained from solving several DIMACS graph instances. Observe that the densities in these graph tend to be smaller than $d = 0.2$, a common trait of real-life graphs. Consistent with previous results for sparse graphs, the combinatorial BnB algorithms outperform CPLEX by one to four orders of magnitude, and the difference between the two BnB algorithm variants was more pronounced in these cases. Indeed, the BnB algorithm employing acyclic path decomposition was between one to two orders of magnitude faster than the one employing approximate coloring.

5. CONCLUSIONS

We have introduced a new class of two-stage stochastic maximum subgraph problems for finding the maximum expected size of a graph that satisfies a defined structural property Π. Emphasis was put on identifying subgraphs whose properties can be restored within a limited repair budget in the presence of structural uncertainties that manifest in the form of random connection (edge) changes/failures. A combinatorial BnB algorithm exploiting the structure of two-stage stochastic maximum Π subgraph problems was developed. Our technique utilizes two combinatorial BnB algorithms for finding optimal first- and second-stage subgraph solutions.

The proposed framework applies to a broad range of graph properties, and in this work, we illustrated the proposed approach on an example where the property of interest Π defines a clique. Numerical simulations on randomly generated graphs and DIMACS graphs indicate that solution times can be reduced by several orders of magnitude via the proposed BnB algorithm in comparison to an equivalent mathematical programming solver. Namely, for all the tested graph configurations other than ones with the high edge density of $d = 0.8$, between one or more orders of magnitude in performance improvements were observed.

Subsequent extensions of the proposed model and solution methods will consider nonhereditary graph properties, particularly in a broader context of identifying cohesive clusters in stochastic networks. It is also of interest to examine extensions that involve variations in structural properties Π between decision stages, reparability cost structures, and budgetary restriction.

ACKNOWLEDGMENTS

This research was performed while the first and second authors held National Research Council Research Associateship Awards at the Air Force Research Laboratory. In addition, support by the AFRL Mathematical Modeling and Optimization Institute is gratefully acknowledged.

REFERENCES

[1] V.E. Alekseev and D. Korobitsyn, Complexity of some problems on hereditary classes of graphs, Diskretnaya Matematika 4 (1992), 34–40.

[2] N. Alon and A. Shapira, A characterization of the (natural) graph properties testable with one-sided error, SIAM J Comput 37 (2008), 1703–1727.

[3] A. Atamtürk and M. Zhang, Two-stage robust network flow and design under demand uncertainty, Oper Res 55 (2007), 662–673.

[4] H.J. Bandelt and H.M. Mulder, Distance-hereditary graphs, J Comb Theory Ser B 41 (1986), 182–208.

[5] J.R. Birge and F. Louveaux, Introduction to stochastic programming, Springer, New York, 1997.

[6] V.L. Boginski, C.W. Commander, and T. Turko, Polynomial-time identification of robust network flows under uncertain arc failures, Optim Lett 3 (2009), 461–473.

[7] I. Bomze, M. Chimani, M. Jünger, I. Ljubić, P. Mutzel, and B. Zey, "Solving two-stage stochastic Steiner tree problems by two-stage branch-and-cut," Algorithms and computation, Vol. 6506 of Lecture Notes in Computer Science, O. Cheong, K.Y. Chwa, and K. Park (Editors), Springer Berlin Heidelberg, Germany, 2010, pp. 427–439.

[8] S.V. Buldyrev, R. Parshani, G. Paul, H.E. Stanley, and S. Havlin, Catastrophic cascade of failures in interdependent networks, Nature 464 (2010), 1025–1028.

[9] A.M. Campbell and B.W. Thomas, Probabilistic traveling salesman problem with deadlines, Transp Sci 42 (2008), 1–21.

[10] R. Carraghan and P.M. Pardalos, An exact algorithm for the maximum clique problem, Oper Res Lett 9 (1990), 375–382.

[11] R.K. Cheung and C.Y. Chen, A two-stage stochastic network model and solution methods for the dynamic empty container allocation problem, Transp Sci 32 (1998), 142–162.

[12] P. Erdös and A. Rényi, On the evolution of random graphs, Publication Math Inst Hungarian Acad Sci 5 (1960), 17–61.

[13] M. Fiedler, Algebraic connectivity of graphs, Czechoslovak Math J 23 (1973), 298–305.

[14] E. Fox-Epstein and D. Krizanc, "The complexity of minor-ancestral graph properties with forbidden pairs," Computer science—Theory and applications, Vol. 7353 of Lecture Notes in Computer Science, E. Hirsch, J. Karhumäki, A. Lepistö, and M. Prilutskii (Editors), Springer Berlin Heidelberg, 2012, pp. 138–147.

[15] A. Ghosh and S. Boyd, Growing well-connected graphs, 45th IEEE Conference Dec Control, San Diego, California, USA, Dec 2006, pp. 6605–6611.

[16] G.D. Glockner and G.L. Nemhauser, A dynamic network flow problem with uncertain arc capacities: Formulation and problem structure, Oper Res 48 (2000), 233–242.

[17] J. Konc and D. Janezic, An improved branch and bound algorithm for the maximum clique problem, MATCH Commun Math Comput Chem 58 (2007), 569–590.

[18] G. Laporte, F.V. Louveaux, and L. van Hamme, An integer *L*-shaped algorithm for the capacitated vehicle routing problem with stochastic demands, Oper Res 50 (2002), 415–423.

[19] I. Ljubić, P. Mutzel, and B. Zey, Stochastic survivable network design problems, Electron Notes Discr Math 41 (2013), 245–252.

[20] Z. Miao, B. Balasundaram, and E.L. Pasiliao, An exact algorithm for the maximum probabilistic clique problem, J Comb Optim 28 (2014), 105–120.

[21] P.R.J. Östergård, A fast algorithm for the maximum clique problem, Discr Appl Math 120 (2002), 197–207, Special Issue devoted to the 6th Twente Workshop on Graphs and Combinatorial Optimization.

[22] P.M. Pardalos and J. Xue, The maximum clique problem, J Global Optim 4 (1994), 301–328.

[23] A. Prékopa, Stochastic programming, Kluwer Academic Publishers, Dordrecht, 1995.

[24] N. Robertson and P. Seymour, Graph minors. I. Excluding a forest, J Comb Theory Ser B 35 (1983), 39–61.

[25] N. Robertson and P. Seymour, Graph minors. XIX. Well-quasi-ordering on a surface, J Comb Theory Ser B 90 (2004), 325–385.

[26] M. Rysz, M. Mirghorbani, P. Krokhmal, and E.L. Pasiliao, On risk-averse maximum weighted subgraph problems, J Comb Optim 28 (2014), 167–185.

[27] E. Tomita and T. Seki, "An efficient branch-and-bound algorithm for finding a maximum clique," Discrete mathematics and theoretical computer science, Vol. 2731 of Lecture Notes in Computer Science, C. Calude, M. Dinneen, and V. Vajnovszki (Editors), Springer Berlin Heidelberg, 2003, pp. 278–289.

[28] P. Tsiakis, N. Shah, and C.C. Pantelides, Design of multi-echelon supply chain networks under demand uncertainty, Indust Eng Chem Res 40 (2001), 3585–3604.

[29] A. Veremyev and V. Boginski, Identifying large robust network clusters via new compact formulations of maximum *k*-club problems, Eur J Oper Res 218 (2012), 316–326.

[30] B. Verweij, S. Ahmed, A.J. Kleywegt, G. Nemhauser, and A. Shapiro, The sample average approximation method applied to stochastic routing problems: A computational study, Comput Optim Appl 24 (2003), 289–333.

[31] S. Voccia, A. Campbell, and B. Thomas, The probabilistic traveling salesman problem with time windows, EURO J Transp Logist 2 (2013), 89–107.

[32] J.W. Wang and L.L. Rong, Robustness of the western United States power grid under edge attack strategies due to cascading failures, Saf Sci 49 (2011), 807–812.

[33] K. Yamaguchi and S. Masuda, A new exact algorithm for the maximum weight clique problem, 23rd Int Conference Circuits/Systems, Comput Commun (ITC-CSCC'08), Shimonoseki City, Yamaguchi, Japan, 2008, pp. 317–320.

[34] M. Yannakakis, Node-and edge-deletion NP-complete problems, STOC'78: Proc 10th Ann ACM Symp Theory Comput, ACM Press, New York, 1978, pp. 253–264.

[35] O. Yezerska, S. Butenko, and V. Boginski, Detecting robust cliques in graphs subject to uncertain edge failures, Ann Oper Res (2016), in press. doi: 10.1007/s10479-016-2161-0.