Mixed Integer Linear Programming Formulations for Robust Surgery Scheduling

Ankit Bansal[*] Jean-Philippe Richard[†] Bjorn P. Berg[‡] Yu-Li Huang[§]

Abstract

We introduce Mixed Integer Linear Programming (MILP) formulations for the two-stage robust surgery scheduling problem (*2SRSSP*). We derive these formulations by modeling the *second-stage* problem as a longest path problem on a layered acyclic graph and subsequently converting it into a linear program. This linear program is then dualized and integrated with the first-stage, resulting in a MILP formulation for *2SRSSP*. Additionally, we propose methods to improve the computational performance of the these MILP formulations. An extensive numerical study, based on data from an academic medical center, reveals that the computational performance of the Column and Constraint Generation (`C&CG`) Algorithm, the only exact method previously applied to this problem in the literature, is outperformed by at least one of the MILP formulations we introduce in this paper.

1 Introduction

Effective operating room (OR) management continues to receive high attention from healthcare delivery systems and the research community (Schouten et al. 2023). ORs represent a substantial portion of healthcare expenses and resources while serving as critical hubs for delivering surgical care. Efficient management of these resources aims to simultaneously attain high capacity utilization and high service levels and access for patients while containing costs. Effective OR management directly contributes to improved healthcare outcomes, efficient use of resources, and the financial sustainability of healthcare delivery systems.

In this paper we introduce a monolithic mixed integer linear programming (MILP) formulation for the two-stage robust surgery scheduling problem (*2SRSSP*). The primary objective of *2SRSSP* is

[*]Department of Systems Science and Industrial Engineering, State University of New York-Binghamton, Binghamton, NY, 13905, abansal@binghamton.edu (*Corresponding author*)

[†]Department of Industrial and Systems Engineering, University of Minnesota, Minneapolis, MN, 55455

[‡]Division of Health Policy and Management, School of Public Health, University of Minnesota, Minneapolis, MN, 55455

[§]Robert D. and Patricia E. Kern Center for the Science of Healthcare Delivery, Mayo Clinic, Rochester, MN, 55905

to minimize the combined expenses related to opening ORs and overtime costs when surgeries extend beyond their initially planned OR time. In the first stage of *2SRSSP*, decisions are made regarding how many ORs to open and how surgeries will be allocated to the opened ORs. Subsequently, the second stage identifies and evaluates the worst-case costs with respect to the first-stage allocation decisions, subject to a budget of uncertainty, which imposes an upper-bound on the number of surgeries that surpass their anticipated duration.

The problem addressed in this paper, *2SRSSP*, was first described by Denton et al. (2010) as a robust extension of a deterministic formulation and a two-stage stochastic programming formulation of the surgery scheduling problem. Exact solution approaches for *2SRSSP* based on Column and Constraint (`C&CG`) algorithms have been proposed (Neyshabouri and Berg 2017, Bansal et al. 2021). Yet, these approaches suffer from long solution times for moderately-sized problems. The bilinear modeling approach typically employed to formulate the *second-stage* of *2SRSSP* has prevented the formulation of a monolithic MILP for *2SRSSP* and the computational benefits such formulations typically confer. The approach that we propose yields new exact monolithic MILP formulations of *2SRSSP* that can be readily solved using commercial solvers.

1.1 Problem statement

In this section we present a general *2SRSSP* formulation based on that which was initially presented by Denton et al. (2010) and subsequently studied in the literature. Consider a surgery-to-OR assignment problem where a collection of surgeries, represented by the set N, are to be assigned to a set of ORs, represented by the set M. We let $m := |M|$ and $n := |N|$. Each surgery i in N has a nominal duration d_i^l and an upper bound on its deviation above the nominal duration d_i°. All ORs in M have capacity U, which represents the maximum amount of time that can be allocated to surgeries in each room without incurring additional overage costs. Opening an OR incurs a fixed cost of c^f. If an OR is utilized beyond its capacity U, an additional cost c° is incurred for each minute it exceeds the capacity.

Binary decision variables $y_{i,j}$ indicate whether surgery $i \in N$ is assigned to OR $j \in M$, whereas binary variables x_j indicate whether OR $j \in M$ is opened. Variable β represents the worst case overtime for a given $(\mathbf{x}, \mathbf{y})$. To ensure robustness in decision-making, a *second-stage* problem is formulated to calculate the worst-case overtime that can result from solution $(\mathbf{x}, \mathbf{y})$. This worst-case overtime is determined by introducing binary variables r_i, which indicate whether surgery i exceeds its nominal time. The number of surgeries that have durations longer than their nominal completion time is upper bounded by τ, which is referred to as the *budget of uncertainty.* Using this notation, the *2SRSSP* is formulated as

$$
\begin{aligned}
v^* := \min \quad & \sum_{j \in M} c^f x_j + c^\circ \beta && && \text{(1a)}\\
\text{s.t.} \quad & \beta \geq F(\mathbf{x}, \mathbf{y}) && && \text{(1b)}\\
& \sum_{j \in M} y_{i,j} = 1 && \forall i \in N && \text{(1c)}\\
& y_{i,j} \leq x_j && \forall i \in N, \forall j \in M && \text{(1d)}\\
& x_j \geq x_{j+1} && \forall j \in M \setminus m && \text{(1e)}\\
& \sum_{j=1}^{k} y_{k,j} = 1 && \forall k \in M && \text{(1f)}\\
& \sum_{k=j}^{\min\{i,m\}} y_{i,k} \leq \sum_{l=j-1}^{i-1} y_{l,j-1} && \forall i \in N, \forall j \in M, i \geq j && \text{(1g)}\\
& \beta \geq 0,\ x_j \in \{0,1\},\ y_{i,j} \in \{0,1\} && \forall i \in N,\ \forall j \in M, && \text{(1h)}
\end{aligned}
$$

where

$$
\begin{aligned}
F(\mathbf{x}, \mathbf{y}) := \max \quad & \sum_{j \in M} \left(\sum_{i \in N} \left(d_i^l + r_i d_i^\circ \right) y_{i,j} - U x_j \right)^+ && && \text{(2a)}\\
\text{s.t.} \quad & \sum_{i \in N} r_i \leq \tau && && \text{(2b)}\\
& r_i \in \{0,1\} && \forall i \in N, && \text{(2c)}
\end{aligned}
$$

where $(\cdot)^+ := \max\{\cdot, 0\}$. Objective function (1a) aims to minimize the total cost, which is composed of the fixed costs of opening ORs and the total worst-case overtime cost that would occur from the choice of $\mathbf{x}$ and $\mathbf{y}$. Constraint (1b) defines the worst-case overtime cost and is satisfied at equality in optimal solutions. Constraints (1c) ensure that each surgery is assigned to exactly one OR whereas constraints (1d) ensure that a surgery can be assigned to an OR only if it is open. Constraints (1e)-(1g) are the symmetry-breaking constraints described in Section 4.1 of Denton et al. (2010).

The robustness of a solution $(\mathbf{x}, \mathbf{y}) \in \{0,1\}^{m+mn}$ is evaluated through (1b) and (2a)-(2c) which determine the maximum possible total overtime cost that can be incurred from solution $(\mathbf{x}, \mathbf{y})$. Objective function (2a) aims to maximize total overtime. Constraints (2b) ensure that the number of surgeries selected to exceed their nominal time does not exceed the specified budget of uncertainty,

τ. In formulating (2), we assume that any surgery exceeding its nominal duration takes its maximum possible completion time since there is always an optimal solution of *2SRSSP* for which this property holds (Neyshabouri and Berg 2017). A solution $(\beta, \mathbf{x}, \mathbf{y}) \in [0, \infty) \times \{0,1\}^{m+mn}$ is said to be *robust feasible* if it satisfies (1b)-(1h) and is said to be *robust optimal* if it is optimal for (1). To maintain brevity, we often omit mentioning the component β when discussing *robust feasible* and *robust optimal* solutions $(\beta, \mathbf{x}, \mathbf{y})$ of (1). However, unless stated otherwise, a solution of (1) denoted as $(\mathbf{x}, \mathbf{y})$ should be understood to have an associated component β that is computed as $F(\mathbf{x}, \mathbf{y})$.

In this paper, we model (2) as a longest-path problem on a layered acyclic graph that, for fixed values of $\mathbf{x}$ and $\mathbf{y}$, leads to a Linear Program (LP) for the computation of $F(\mathbf{x}, \mathbf{y})$. This formulation can then be dualized and integrated with (1) to obtain an exact monolithic MILP reformulation for *2SRSSP*.

1.2 Literature review

OR and surgery management decisions are typically categorized in a hierarchical fashion including strategic decisions (*e.g.*, facility size and types of surgeries), tactical decisions (*e.g.*, scheduling windows and specialty block assignments), and operational decisions (*e.g.*, managing delays and mitigating overtime); see (Gupta 2007). The problem in this paper spans across tactical and operational decisions where managers must decide how many ORs are to be prepared and how surgeries should be assigned to the ORs for a specialty on a particular day. The objective of this problem balances a trade-off between the costs of opening additional ORs to which surgeries are assigned and the costs of using overtime to complete the day's scheduled surgeries. Surgery duration uncertainty is directly considered in this problem, although demand uncertainty is not as we focus on scheduling specialty elective cases. This model was initially proposed and studied by Denton et al. (2010). The authors reformulate the second stage mixed-integer nonlinear program as a mixed-integer linear program and apply LP duality to the resulting LP relaxation to yield an easy to solve, though approximate and not exact, formulation of *2SRSSP* (Ardestani-Jaafari and Delage 2021).

A similar problem is the surgical case assignment problem (SCAP) where surgeries are assigned to operating room blocks across a planning horizon of multiple days. Besides the duration of the planning horizon, a significant difference between SCAP and *2SRSSP* is that in SCAP surgeries are selected from a wait list and a penalty is incurred for delaying care. In contrast, surgeries in *2SRSSP* have been assigned a day and delaying surgery is not an option. This focus gives SCAP the additional benefit of having a linear objective in the second stage. Addis et al. (2014) and Marques and Captivo (2017) both consider two-stage robust formulations of SCAP. While each formulation considers different details of SCAP, both robust formulations rely on applying duality and linearizing the second stage to formulate a mixed-integer linear program.

Rath et al. (2017) develop a variation of *2SRSSP* where anesthesiologists are also assigned to surgeries. As a result the sequence of surgeries within an OR must also be determined in the first stage. Through strong duality the second stage LP is converted, along with the first stage, into a single mixed-integer program. However, the resulting mixed-integer program remains computationally challenging due, in part, to the use of Big-M type constraints. The relatively complete recourse structure of the problem is leveraged in a heuristic algorithm that is used to obtain solutions.

Neyshabouri and Berg (2017) propose a formulation of *2SRSSP* which also considers downstream recovery resource capacity. Surgeries are assigned to OR blocks in the first stage according to specialty. The second stage involves two separate recourse problems where in the first the worst-case overtime costs are determined and in the second the worst-case costs associated with insufficient recovery capacity are determined. Both recourse problems have max-min objective structures and are reformulated into mixed-integer linear programs. A `C&CG` algorithm is adapted from the algorithm presented in Zeng and Zhao (2013) to obtain exact solutions.

A reformulation of *2SRSSP* as a value function-based formulation is presented by Bansal et al. (2021). The resulting formulation's structure is analyzed to identify cuts that strengthen the formulation. The value function-based formulation is solved using a `C&CG` algorithm. Computation results show an improvement with the proposed formulation when compared to that in Neyshabouri and Berg (2017). The second stage of *2SRSSP* was approximated using a series of valid inequalities by Bansal et al. (2024). The valid inequalities yield an LP relaxation that more closely approximates the original second stage mixed-integer program. The duality of the resulting LP is then used to reformulate *2SRSSP* as a single mixed-integer linear program. This formulation is approximate and not exact. Computational experiments show favorable results, when compared to those obtained with `C&CG`.

1.3 Contributions

The contributions of this paper are as follows:

1. We present a new formulation of the *second-stage* problem, (2) which is an LP for fixed values of $(\mathbf{x}, \mathbf{y})$. We do so by modeling the *second-stage* problem as longest path problem on a layered acyclic graph.

2. We derive new monolithic exact MILP formulations for *2SRSSP* by dualizing the LP formulation for the *second-stage* problem and integrating with the first-stage problem.

3. We propose methods to enhance the computational performance of these MILP formulations. These methods include strengthening the constraints, introducing valid inequalities, integrating these formulations with `C&CG` formulation for *2SRSSP* (Neyshabouri and Berg 2017, Zeng and Zhao 2013), and deriving a lower bound for the optimal objective function of *2SRSSP*.

The remainder of the paper is organized as follows. In Section 2, we introduce a longest path formulation for the *second-stage* problem. Section 3 covers the derivation of the exact MILP formulations for *2SRSSP*, including methods to strengthen the constraints, and introduces valid inequalities for these formulations. In Section 4, we integrate the formulations from Section 3 with the `C&CG` formulation for *2SRSSP*, resulting in a new formulation that demonstrates improved computational results in our experiments. Section 5 presents the derivation of a lower bound for the optimal objective function value of *2SRSSP*. Section 6 details the computational experiments and their results. We conclude the paper in Section 7.

2 A longest path formulation for the *second-stage* problem

Given a leader solution $(\mathbf{x}, \mathbf{y})$, we next formulate the problem of computing $F(\mathbf{x}, \mathbf{y})$ as a longest path problem on a layered acyclic digraph that we refer to as $G^{\mathbf{x},\mathbf{y}}(V, E)$. We define $L = \{0, \ldots, \tau\}$. The node set V contains $m(\tau + 1) + 2$ nodes. In addition to a source node s and a sink node t, the digraph has m internal layers (one for each OR), each containing $\tau + 1$ nodes. We denote the nodes in the internal layers as $v_{j,k}$, where $j \in M$ is the index of the layer/OR and $k \in L$ is the level of the node within its layer. To be consistent with this notation, the source s can be written as $v_{0,0}$ whereas the sink t can be denoted by $v_{m+1,\tau}$. We make use of both notations in the remainder of the paper. The arc set E contains $2(\tau+1)+(m-1)\frac{(\tau+1)(\tau+2)}{2}$ arcs. Arcs $(s, v_{1,k})$, for $k \in L$, are created between s and each node of the first layer. Arcs are created between each node $v_{j,k}$ of an internal layer $j \in M\backslash\{m\}$, and each node $v_{j+1,k'}$ of the following layer that has higher level $k' \geq k$. These arcs are of the form $(v_{j,k}, v_{j+1,k'})$. Finally, arcs $(v_{m,k}, t)$, for $k \in L$, are created between each node of the last internal layer and the sink node t. An illustration of $G^{\mathbf{x},\mathbf{y}}(V, E)$ is given in Figure 1. Because arcs only go from one layer to the next, digraph $G^{\mathbf{x},\mathbf{y}}(V, E)$ is acyclic. Paths from s to t in this digraph correspond to the process of allocating extensions of surgery durations between ORs. In particular, node $v_{j,k}$ is visited when at total of k extensions have been assigned to ORs $1, \ldots, j$ and arc $(v_{j-1,k}, v_{j,k'})$ is traversed when assigning an additional $k' - k$ extensions to OR j.

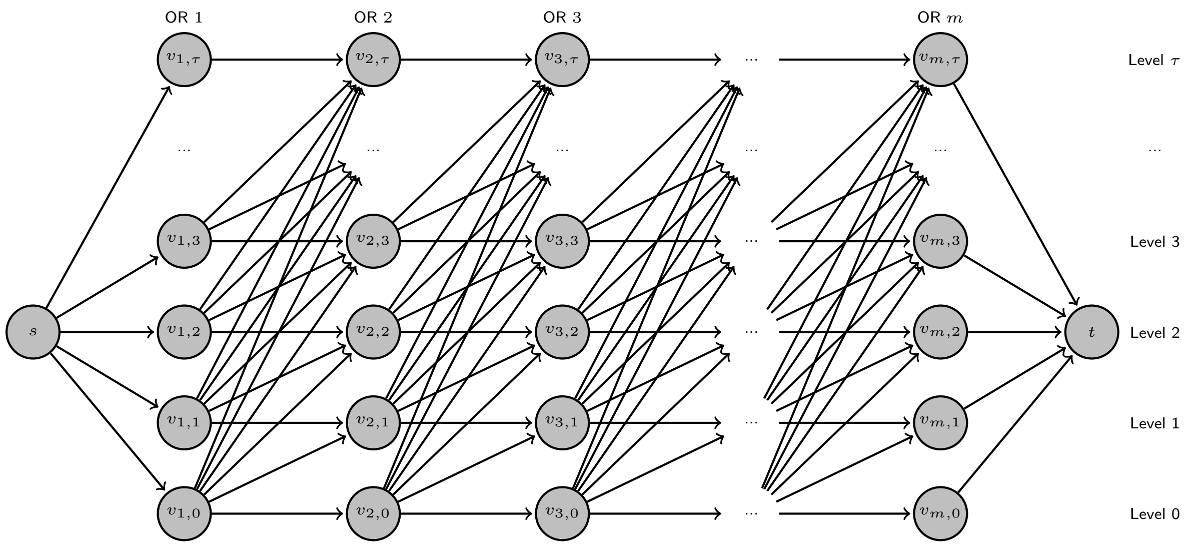

Figure 1: Graph $G^{\mathbf{x},\mathbf{y}}(V, E)$ for the *second-stage* problem

Next, we describe the lengths of the arcs of the digraph. To this end, for $k \in L$ and for any vector $\mathbf{v} \in \mathbb{B}^n$, we introduce the function

$$\mu_k[\mathbf{v}] := \max\left\{ \sum_{i \in N} d_i^\circ v_i \chi_i \;\middle|\; \sum_{i \in N} \chi_i \leq k,\ \ 0 \leq \chi_i \leq 1, \forall i \in N \right\}. \tag{3}$$

Function $\mu_k[\mathbf{v}]$ is the optimal value of an LP that always has an optimal binary solution $\boldsymbol{\chi} = (\chi_i)_{i \in N}$ since its constraint matrix is totally unimodular. Thinking of $\mathbf{v} = (v_i)_{i \in N}$ as an indicator vector of a collection of surgeries collectively assigned to an OR, function $\mu_k[\mathbf{v}]$ computes the sum of the highest k values of d_i° among the surgeries assigned to that OR, *i.e.*, the maximum additional time beyond the nominal duration that is taken when at most k of these surgeries extend beyond their nominal surgery times. Arc lengths in $G^{\mathbf{x},\mathbf{y}}(V, E)$ are then computed as follows:

- The length of arc $(v_{j-1,k}, v_{j,k'})$ where $j \in M \backslash \{m\}$, and $k, k' \in L$ with $k' \geq k$, , which we denote as $c_{j,k,k'}(\mathbf{x}, \mathbf{y})$, is the maximum possible increment in total overtime when at most $k' - k$ surgeries in OR j take their maximum possible time. Specifically,

$$c_{j,k,k'}(\mathbf{x}, \mathbf{y}) = \left\{ \begin{array}{ll} \displaystyle\sum_{i \in N} d_i^l y_{i,j} - U x_j + \mu_{k'-k}[\mathbf{y}_j] & \text{if } k' - k > 0 \\ \max\left\{ 0, \displaystyle\sum_{i \in N} d_i^l y_{i,j} - U x_j \right\} & \text{if } k' - k = 0 \end{array} \right\} \tag{4}$$

where $\max\left\{0, \sum_{i \in N} d_i^l y_{i,j} - U x_j\right\}$ calculates the total overtime in OR j under the condition that none of the surgeries assigned to that OR take their maximum possible time.

- The length of arc $(s, v_{1,k'}) = (v_{0,0}, v_{1,k'})$ where $k' \in L$ is given by $c_{1,0,k'}(\mathbf{x}, \mathbf{y})$, as defined in (4).
- The length of arc $(v_{m,k}, t) = (v_{m,k}, v_{m+1,\tau})$ where $k \in L$, denoted by $c_{m+1,k,\tau}(\mathbf{x}, \mathbf{y})$, is set to 0.

Under this setup, the length of a dipath s, v_{1,k_1}, …, v_{j,k_j} in $G^{\mathbf{x},\mathbf{y}}(V, E)$ from node s to node v_{j,k_j} at level k_j of OR j gives the maximum sum of overtimes among the first j ORs when k_1, $k_2 - k_1$, …, $k_j - k_{j-1}$ surgeries assigned to ORs $\{1, \dots, j\}$ take their maximum possible duration. This observation leads to the following proposition.

Proposition 1. *For* 2SRSSP *with n surgeries, m ORs, budget of uncertainty, τ and for a solution $(\mathbf{x}, \mathbf{y}) \in \mathbb{B}^m \times \mathbb{B}^{n\times m}$ that satisfies* (1c)-(1g)*, the optimal value of the* second-stage *problem* (2) *is equal to the length of the longest dipath from s to t in $G^{\mathbf{x},\mathbf{y}}(V, E)$.*

Proof. Let $(\mathbf{x}, \mathbf{y}) \in \mathbb{B}^m \times \mathbb{B}^{n\times m}$ be a solution that satisfies (1c)-(1g). Let $c^*(\mathbf{x}, \mathbf{y})$ be the length of a longest dipath from s to t in $G^{\mathbf{x},\mathbf{y}}(V, E)$. We show that $c^*(\mathbf{x}, \mathbf{y}) = F(\mathbf{x}, \mathbf{y})$.

First, consider any dipath P from s to t in $G^{\mathbf{x},\mathbf{y}}(V, E)$. We argue that its length $c(P) \le F(\mathbf{x}, \mathbf{y})$. The arcs of P must be (v_{0,k_0}, v_{1,k_1}), (v_{1,k_1}, v_{2,k_2}), …, $(v_{m-1,k_{m-1}}, v_{m,k_m})$, $(v_{m,k_m}, v_{m+1,k_{m+1}})$ for suitable indices k_0, k_1, …, k_m, k_{m+1} such that $0 = k_0 \le k_1 \le \dots \le k_m \le k_{m+1} = \tau$. Define $Z^> = \{j \in M \,|\, k_{j-1} < k_j\}$ and $Z^0 = \{j \in M \,|\, k_{j-1} = k_j\}$. Then, the length of P is given by $c(P) = \sum_{j\in Z^>} \left(\sum_{i\in N} d^l_i y_{i,j} - Ux_j + \mu_{k_j - k_{j-1}}\left[\mathbf{y}_j\right]\right) + \sum_{j\in Z^0} \left(\max\left\{0, \sum_{i\in N} d^l_i y_{i,j} - Ux_j\right\}\right)$. Without loss of generality we assume that, for each $j \in Z^>$, $c_j := \sum_{i\in N} d^l_i y_{i,j} - Ux_j + \mu_{k_j-k_{j-1}}\left[\mathbf{y}_j\right] > 0$ as otherwise, a longer (or an equal length) dipath can be obtained by setting $k'_0 = 0$, $k'_{m+1} = \tau$, and, for $j \in M$, $k'_j = k'_{j-1}+(k_j-k_{j-1})$ for $j \in Z^>$ with $c_j > 0$ and $k'_j = k'_{j-1}$ otherwise. Further, for each $j \in M$, let $\chi^{[j]}$ be an optimal vertex of the problem defining $\mu_{k_j-k_{j-1}}\left[\mathbf{y}_j\right]$ in (3). We may assume, without loss of generality, that this solution is such that (i) $\sum_{i\in N} \chi^{[j]}_i \le k_j-k_{j-1}$ for each $j \in M$, and (ii) $\chi^{[j]}_i \le y_{i,j}$ for each $i \in N$ and $j \in M$. For each $i \in N$, define $r^*_i := \sum_{j\in M} \chi^{[j]}_i \le \sum_{j\in M} y_{i,j} = 1$. It follows that $\sum_{i\in N} r^*_i = \sum_{j\in M}\sum_{i\in N} \chi^{[j]}_i \le \sum_{j\in M}(k_j - k_{j-1}) = k_m - k_0 \le \tau$. This ensures that $\mathbf{r}^*$ satisfies (2b). Using this definition of $\mathbf{r}^*$, we obtain that $c(P) = \sum_{j\in Z^>} \left(\left(\sum_{i\in N} d^l_i y_{i,j} - Ux_j\right) + \sum_{i\in N} d^\circ_i y_{i,j} r^*_i\right) + \sum_{j\in Z^0} \left(\left(\sum_{i\in N} d^l_i y_{i,j} - Ux_j\right)^+\right) = \sum_{j\in M} \left(\sum_{i\in N} \left(d^l_i + r^*_i d^\circ_i\right) y_{i,j} - Ux_j\right)^+ \le F(\mathbf{x}, \mathbf{y})$.

Second, consider an optimal solution $\mathbf{r}^*$ of (2) and define $c_j := \left(\left(\sum_{i\in N} \left(d^l_i + r^*_i d^\circ_i\right) y_{i,j} - Ux_j\right)^+\right)$. Then, $F(\mathbf{x}, \mathbf{y}) = \sum_{j\in M} c_j$. Next, define $Z^> = \{j \in M \,|\, c_j > 0\}$ and $Z^0 = \{j \in M \,|\, c_j = 0\}$. Without loss of generality, we assume that for each $i \in N$ such that $y_{i,j} = 1$ for some $j \in Z^0$, we have that $r^*_i = 0$. This is because there is no reason to extend the duration of surgery i in OR j if no overtime occurs when this surgery is extended. For each $j \in Z^>$, define $\eta_j = \{i \in N \,|\, r^*_i = 1, y_{i,j} = 1\}$. Next, set $k_0 = 0$, $k_j = k_{j-1}$ for each $j \in Z^0$, and $k_j = k_{j-1} + |\eta_j|$ for each $j \in Z^>$. As $\mathbf{r}^*$ is an optimal solution of (2), $\sum_{i\in N} d_i^\circ r_i^* y_{i,j} = \mu_{|\eta_j|}\left[\mathbf{y}_j\right]$ for each $j \in Z^>$. Now consider a dipath P of $G^{\mathbf{x},\mathbf{y}}(V,E)$ with arcs (v_{0,k_0}, v_{1,k_1}), (v_{1,k_1}, v_{2,k_2}), $\ldots$, $(v_{m-1,k_{m-1}}, v_{m,k_m})$, $(v_{m,k_m}, v_{m+1,k_{m+1}})$. Let $c(P)$ be the length of P. With the notations introduced above, we write that

$$\begin{aligned}
F(\mathbf{x},\mathbf{y}) &= \sum_{j\in Z^>}\left(\sum_{i\in N}\left(d_i^l + r_i^* d_i^\circ\right) y_{i,j} - U x_j\right)\\
&= \sum_{j\in Z^>}\left(\sum_{i\in N} d_i^l y_{i,j} - U x_j + \mu_{|\eta_j|}\left[\mathbf{y}_j\right]\right)\\
&= \sum_{j\in Z^>}\left(\sum_{i\in N} d_i^l y_{i,j} - U x_j + \mu_{k_j - k_{j-1}}\left[\mathbf{y}_j\right]\right)\\
&= \sum_{j\in Z^>}\left(\sum_{i\in N} d_i^l y_{i,j} - U x_j + \mu_{k_j - k_{j-1}}\left[\mathbf{y}_j\right]\right) + \sum_{j\in Z^0}\max\left(0, \left(\sum_{i\in N} d_i^l y_{i,j} - U x_j\right)\right)\\
&= c(P)\\
&\le c^*(\mathbf{x},\mathbf{y})
\end{aligned}$$

where the second to last equality is due to the definition of Z^0 and dipath P, and the last equality holds because of (4).

□

As a consequence of Proposition 1, we can replace the formulation (2) for the *second-stage* problem with the problem of determining the longest path between s and t in the layered acyclic digraph $G^{\mathbf{x},\mathbf{y}}(V,E)$. It is well-known that such a longest path problem can be formulated as an LP; see (Ahuja et al. 1988). We give an explicit description of this LP next.

Proposition 2. *Let* $(\mathbf{x},\mathbf{y}) \in \mathbb{B}^m \times \mathbb{B}^{n\times m}$ *be a solution that satisfies* (1c)-(1d). *Then*

$$F(\mathbf{x},\mathbf{y}) = \max \sum_{k\in L} c_{1,0,k}(\mathbf{x},\mathbf{y})\, w_{1,0,k} + \sum_{j\in M\setminus\{1\}}\sum_{k\in L}\sum_{l\in L: l\ge k} c_{j,k,l}(\mathbf{x},\mathbf{y})\, w_{j,k,l} \tag{5a}$$

$$[\pi_{0,0}] \qquad s.t. \quad -\sum_{k\in L} w_{1,0,k} = -1 \tag{5b}$$

$$[\pi_{1,k}] \qquad w_{1,0,k} - \sum_{l\in L: l\ge k} w_{2,k,l} = 0 \qquad \forall k \in L \tag{5c}$$

$$[\pi_{j,k}] \qquad \sum_{k'\in L: k'\le k} w_{j,k',k} - \sum_{l\in L: l\ge k} w_{j+1,k,l} = 0 \qquad \forall k \in L, \forall j \in M\setminus\{1,m\} \tag{5d}$$

$$[\pi_{m,k}] \qquad \sum_{k'\in L: k'\le k} w_{m,k',k} - w_{m+1,k,\tau} = 0 \qquad \forall k \in L \tag{5e}$$

$$[\pi_{m+1,\tau}] \qquad \sum_{k\in L} w_{m+1,k,\tau} = 1 \tag{5f}$$

$$w_{j,k,l} \ge 0 \qquad \forall j \in M\setminus\{1\}, \forall k,l \in L : l \ge k \tag{5g}$$

$$w_{1,0,k} \geq 0 \;\; w_{m+1,k,\tau} \geq 0, \qquad\qquad \forall k \in L. \tag{5h}$$

Consequently, the value of $F(\mathbf{x}, \mathbf{y})$ *can be computed through the solution of an LP for any given* $(\mathbf{x}, \mathbf{y})$.

Proof. It is well-known that longest path problems in acyclic digraphs can be formulated as minimum cost flow problems where a unit of supply is provided at the source node s and a unit of demand is extracted at the sink node t. Minimum cost flows problems are formulated using variables to indicate the amount of flow on each arc of the digraph and flow balance equations to require that the flow into a node matches the flow out of a node. The constraint matrix generated by the flow balance constraints is the incidence matrix of the digraph, which is known to be totally unimodular.

To formulate this problem, we define variables $w_{1,0,k}$ for $k \in L$, to represent the total flow from source node $s = v_{0,0}$ to node $v_{1,k}$ at level k of OR 1. Similarly, for $j \in M \backslash \{1\}$ and $k, l \in L$ with $l \geq k$, we define variables $w_{j,k,l}$ to represent the total flow from node $v_{j-1,k}$ at level k of OR $j-1$ to node $v_{j,l}$ at level l of OR j. Finally, for $k \in L$, we define variables $w_{m+1,k,\tau}$ to represent the total flow from node $v_{m,k}$ at level k of OR m to sink node $t = v_{m+1,\tau}$. The flow balance constraints for $G^{\mathbf{x},\mathbf{y}}(V, E)$ are as follows: (5b) ensures that one unit of flow is sent from s, (5f) ensures that one unit of flow arrives at t, and (5c)-(5e) are flow balance constraints for nodes $v_{1,k}$, $v_{j,k}$, and $v_{m,k}$, respectively. Finally, objective (5a) maximizes the total distance of a unit flow traveling from s to t in $G^{\mathbf{x},\mathbf{y}}(V, E)$ by multiplying the flow on each arc by its length. □

LP (5) for the *second-stage* problem can be dualized and integrated with (1) to obtain a monolithic MILP model for *2SRSSP*. Additionally, this LP captures the value of $F(\mathbf{x}, \mathbf{y})$ exactly, and thus leading to a monolithic MILP that is an exact reformulation of *2SRSSP*. We state the dual variables of the constraints in square brackets before each constraint in (5). We formalize this discussion in the next section.

3 MILP formulations for *2SRSSP*

In this section, we present two MILP formulations for *2SRSSP*. These formulations are derived in two steps. In the first, we integrate the dual of the shortest path formulation (5) of the second stage (2) within (1). This formulation makes use of the functions $\mu_k[\mathbf{v}]$ defined in (3). In the second step, we introduce two approaches to represent $\mu_k[\mathbf{v}]$ with linear constraints.

Proposition 3. *The following optimization model is an exact formulation for* 2SRSSP*:*

$$
\begin{aligned}
z^* = \min \quad & \sum_{j \in M} c^f x_j + c^\circ \beta && && \text{(6a)}\\
s.t. \quad & \text{(1c)} - \text{(1h)} && && \\
& \beta \geq \pi_{m+1,\tau} && && \text{(6b)}\\
& \pi_{1,0} \geq 0 && && \text{(6c)}\\
& \pi_{1,0} \geq \sum_{i \in N} d_i^l y_{i,1} - U x_1 && && \text{(6d)}\\
& \pi_{1,k} \geq \sum_{i \in N} d_i^l y_{i,1} - U x_1 + \mu_k \left[\mathbf{y}_1\right] && \forall k \in L \backslash \{0\} && \text{(6e)}\\
& \pi_{j,k} - \pi_{j-1,k} \geq 0 && \forall j \in M \backslash \{1\}, \forall k \in L && \text{(6f)}\\
& \pi_{j,k} - \pi_{j-1,k} \geq \sum_{i \in N} d_i^l y_{i,j} - U x_j && \forall j \in M \backslash \{1\}, \forall k \in L && \text{(6g)}\\
& \pi_{j,l} - \pi_{j-1,k} \geq \sum_{i \in N} d_i^l y_{i,j} - U x_j + \mu_{l-k} \left[\mathbf{y}_j\right] && \forall j \in M \backslash \{1\}, \forall k \in L : l \geq k+1 && \text{(6h)}\\
& \pi_{m+1,\tau} - \pi_{m,k} \geq 0 && \forall k \in L. && \text{(6i)}
\end{aligned}
$$

Proof. Using the formulation of $F(\mathbf{x}, \mathbf{y})$ given in (5) and linear programming duality, constraint (1b) of the leader problem can be rewritten as

$$
\begin{aligned}
& \beta \geq \pi_{m+1,\tau} - \pi_{0,0} && && \text{(7a)}\\
& \pi_{1,k} - \pi_{0,0} \geq c_{1,0,k}(\mathbf{x}, \mathbf{y}) && \forall k \in L && \text{(7b)}\\
& \pi_{j,l} - \pi_{j-1,k} \geq c_{j,k,l}(\mathbf{x}, \mathbf{y}) && \forall j \in M \backslash \{1\}, \forall k, l \in L : l \geq k && \text{(7c)}\\
& \pi_{m+1,\tau} - \pi_{m,k} \geq 0 && \forall k \in L. && \text{(7d)}
\end{aligned}
$$

Because the constraints in (5) are linearly dependent (since the rows of the incidence matrix of a digraph sum up to the zero vector), we may set any of the dual variables $\pi_{j,k}$ to zero. Replacing the arc lengths $c_{1,0,k}(\mathbf{x}, \mathbf{y})$ and $c_{j,k,l}(\mathbf{x}, \mathbf{y})$ in (7b) and (7c) by their definitions in (4) and fixing $\pi_{0,0} = 0$, (7b) yields (6c), (6d) and (6e), and (7c) yields (6f), (6g) and (6h). Finally, (6i) is the same as (7d). □

Function $\mu_k \left[\mathbf{y}_j\right]$ in (6e) and (6h) is defined as the maximum objective function value of the LP (3) when $\mathbf{v} = \mathbf{y}_j$. The following lemma introduces two alternative methods for representing $\mu_k \left[\mathbf{y}_j\right]$ that we will use to develop two explicit MILP formulations for *2SRSSP*.

Lemma 1. *For $k = 0, \dots, \tau$, it holds that*

$$\mu_k\left[\mathbf{y}_j\right] = \min\left\{ k\rho_{j,k} + \sum_{i\in N} \sigma_{j,k,i} \;\middle|\; \rho_{j,k} + \sigma_{j,k,i} \geq d_i^\circ y_{i,j}, \forall i \in N, \rho_{j,k}, \sigma_{j,k,i} \geq 0, \forall i \in N \right\} \tag{8a}$$

$$\mu_k\left[\mathbf{y}_j\right] = \max_{S\subseteq N : |S| \leq k} \left\{ \sum_{i\in S} d_i^\circ y_{i,j} \right\}. \tag{8b}$$

Proof. The LP in (8a) is the dual of the LP in (3) for $\mathbf{v} = \mathbf{y}_j$ where $\rho_{j,k}$ and $\sigma_{j,k,i}$ $i \in N$ are the dual variables of the cardinality and bound constraints in (3), respectively. As the LP in (3) is feasible and bounded, the equality in (8a) holds due the Strong Duality Theorem. The maximization problem described on the right side of equation (8b) aims to select a subset of N consisting of no more than k surgeries assigned to OR j, that maximizes the sum of d_i° values. Thus, the optimal objective function value of this maximization problem must be equal to $\mu_k\left[\mathbf{y}_j\right]$. □

We use Lemma 1 to derive two MILP reformulations of (6). Section 3.1 gives an MILP formulation obtained by replacing (6e) and (6h) using inequalities derived from (8a). Section 3.2 presents another MILP formulation, which is obtained by replacing (6e) and (6h) using inequalities derived from (8b).

3.1 Reformulation of MIP (6) using (8a)

A first formulation is obtained by using the expression for $\mu_k\left[\mathbf{y}_j\right]$ stated in (8a) to reformulate (6e) and (6h). This formulation requires the introduction of variables $\rho_{j,k}$ and $\sigma_{j,k,i}$, which are polynomially many in m, n, and τ. We obtain the following result.

Proposition 4. *Model (vF) defined as*

$$\begin{aligned}
v^* = \min \quad & \sum_{j\in M} c^f x_j + c^\circ \beta \\
s.t. \quad & (1c) - (1h), \\
& (6b) - (6d), (6f), (6g), (6i) \\
& \pi_{1,k} \geq \sum_{i\in N} d_i^l y_{i,1} - Ux_1 + k\rho_{1,k} + \sum_{i\in N} \sigma_{1,k,i} \quad \forall k \in L\backslash\{0\} && (9a) \\
& \pi_{j,l} - \pi_{j-1,k} \geq \sum_{i\in N} d_i^l y_{i,j} - Ux_j \\
& \qquad + (l-k)\rho_{j,l-k} + \sum_{i\in N} \sigma_{j,l-k,i} \quad \forall j \in M\backslash\{1\}, \forall k \in L : l \geq k+1 && (9b) \\
& \rho_{j,k} + \sigma_{j,k,i} \geq d_i^\circ y_{i,j} \quad \forall i \in N, j \in M, k \in L\backslash\{0\} && (9c) \\
& \rho_{j,k}, \sigma_{j,k,i} \geq 0 \quad \forall i \in N, \forall j \in M, \forall k \in L\backslash\{0\}, && (9d)
\end{aligned}$$

is a valid monolithic reformulation of 2SRSSP*.*

3.2 Reformulation of MIP (6) using (8b)

A second formulation is obtained by using the expression for $\mu_k\left[\mathbf{y}_j\right]$ stated in (8b) to reformulate (6e) and (6h). This formulation requires the introduction of $m\tau$ additional variables $z_{j,k}$, for $j \in M$, $k \in L\backslash\{0\}$ where $z_{j,k} = \mu_k\left[\mathbf{y}_j\right]$ represents the sum of the highest k values of d_i° among the surgeries assigned to OR j. It also linearizes the expression $\mu_k\left[\mathbf{y}_j\right] = \max_{S\subseteq N}\sum_{i\in S} d_i^{\circ} y_{i,j}$ for all $j \in M$ and $k \in L\backslash\{0\}$ by enumerating all possible choices of $S \subseteq N$ with $|S| = k$. We obtain

Proposition 5. *Model (*`zF`*) defined as*

$$
\begin{aligned}
v^* = \min \quad & \sum_{j\in M} c^f x_j + c^{\circ}\beta && && \text{(10a)}\\
s.t. \quad & (1c) - (1h) && && \text{(10b)}\\
& (6b) - (6d), (6f), (6g), (6i) && && \\
& \pi_{1,k} \geq \sum_{i\in N} d_i^l y_{i,1} - Ux_1 + z_{1,k} && \forall k \in L\backslash\{0\} && \text{(10c)}\\
& \pi_{j,l} - \pi_{j-1,k} \geq \sum_{i\in N} d_i^l y_{i,j} - Ux_j + z_{j,l-k} && \forall j \in M\backslash\{1\}, \forall k \in L : l \geq k+1 && \text{(10d)}\\
& z_{j,k} \geq \sum_{i\in S} d_i^{\circ} y_{i,j} && \forall S \subseteq N : |S| = k, \forall k \in L\backslash\{0\}, \forall j \in M && \text{(10e)}\\
& z_{j,k} \geq 0 && \forall k \in L\backslash\{0\}, \forall j \in M, && \text{(10f)}
\end{aligned}
$$

is a valid monolithic reformulation of 2SRSSP*.*

It is worth noting that the number of constraints (10e) in `zF` is exponential in the number of surgeries. Algorithm 1 gives a polynomial time algorithm for separating these inequalities from a solution for which the entries of $\mathbf{y}_j$ are binary. We implement this algorithm to add inequalities (10e) to `zF` in a *lazy* manner using callbacks in a commercial MILP solver.

Algorithm 1 Separation algorithm for (10e)

```
1: Input: A vector $(\mathbf{y}_j^*, z_{j,k}^*)$ where $\mathbf{y}_j^* \in \mathbb{B}^n$ for $j \in M$ and $k \in L\backslash\{0\}$.
2: Output: An inequality of the form (10e) violated by $(\mathbf{y}_j^*, z_{j,k}^*)$, if one exists.
3: $I_j \leftarrow argsort\{d_i^\circ | y_{i,j}^* = 1\}$ Note: sorting is done in non-increasing order    ▷ Step 1
4: Let $I_j = \{i_1, i_2, \dots, i_{\bar{k}}\}$
5: $\tilde{k} \leftarrow \min(k, \bar{k})$    ▷ Step 2
6: if $z_{j,k}^* < \sum_{p=1}^{\tilde{k}} d_{i_p}^\circ$ then    ▷ Step 3
7:     if $(\tilde{k} = k)$ then
8:         return $z_{j,k} \geq \sum_{p=1}^{\tilde{k}} d_{i_p}^\circ y_{i_p,j}$.    ▷ Step 3a
9:     else
10:        return $z_{j,k} \geq \sum_{p=1}^{\tilde{k}} d_{i_p}^\circ y_{i_p,j} + \sum_{i \in S} d_i^\circ y_{ij}$ for any $S \subseteq N \setminus I_j$ with $|S| = k - \tilde{k}$.    ▷ Step 3b
11:    end if
12: else
13:    return No violated inequality (10e) exists for $(\mathbf{y}_j^*, z_{j,k}^*)$ for the given $j$ and $k$.    ▷ Step 4
14: end if
```

For a given OR j, a given k, and a vector $(\mathbf{y}_j^*, z_{j,k}^*)$ where $\mathbf{y}_j^* \in \mathbb{B}^n$, Step 1 of Algorithm 1 involves creating the set I_j of surgeries assigned to OR j, arranged in non-increasing order of their d_i° values. In Step 2, $\tilde{k}$ is computed to represent the maximum number of surgeries assigned to OR j that can be extended to their maximum possible time. This value is upper bounded by k and the total number of surgeries assigned to OR j, $\bar{k}$. Next, if $z_{j,k}^*$ is less than the sum of the d_i° values for the first $\tilde{k}$ surgeries in I_j, then Step 3 considers two cases. If $\tilde{k} = k$, then Step 3a returns the inequality $z_{j,k} \geq \sum_{p=1}^{\tilde{k}} d_{i_p}^\circ y_{i_p,j}$, which is violated by $(\mathbf{y}_j^*, z_{j,k}^*)$. If $\tilde{k} < k$, then Step 3b returns the inequality $z_{j,k} \geq \sum_{p=1}^{\tilde{k}} d_{i_p}^\circ y_{i_p,j} + \sum_{i \in S} d_i^\circ y_{ij}$ for a suitable set $S \subseteq N \setminus I_j$. The second term on the right-hand side of this inequality ensures that all returned inequalities through this algorithm are of type (10e). If $z_{j,k}^*$ is not less than the sum of the d_i° values for the first $\tilde{k}$ surgeries, Step 4 states that no inequality (10e) exists that violates $(\mathbf{y}_j^*, z_{j,k}^*)$. When implementing Algorithm 1 in practice, various choices of S can be considered. Because of the symmetry-breaking constraints (1f) the vectors $(\mathbf{y}_j^*, z_{j,k}^*)$ we consider will always be such that OR j is only assigned surgeries from the set $\{j, j+1, \dots, n\}$. Hence, we will construct S by selecting in priority $k - \tilde{k}$ indices from $\{j, j+1, \dots, n\} \setminus I_j$ with largest values of d°.

The remainder of this section is focused on strengthening `zF`. In Section 3.2.1, we strengthen constraints (10e) using the *Lovász Extension* of $\mu_k[\cdot]$. In Section 3.2.2, we describe valid inequalities

(VIs) that are derived from the definitions of $z_{j,k}$.

3.2.1 Strengthening (10e) using the *Lovász Extension* of $\mu_k[\cdot]$

Inequalities (10e) are used to linearize the constraints

$$z_{j,k} \geq \mu_k\left[\mathbf{y}_j\right], \quad \forall k \in L \backslash \{0\}, \forall j \in M. \tag{11}$$

Although the linearization in (10e) provides the exact value of $z_{j,k}$ when $\mathbf{y}_j$ is binary, its linear relaxation can be improved. A possible approach to improve the relaxation is to replace (11) with constraints of the form

$$z_{j,k} \geq \underset{[0,1]^n}{\operatorname{coe}}\left(\mu_k[\cdot]\right), \tag{12}$$

where $\operatorname{coe}_H(f)$ denotes the convex envelope of function f over domain H. The convex envelope of function f over H is the largest convex function defined over H that lies below f. Although the problem of determining the convex envelope of a given function can be difficult, it is not so for submodular functions, which motivate the following result.

Proposition 6. *For each $k \in L$, $\mu_k[\cdot] : \mathbb{B}^n \to \mathbb{R}_+$ is a submodular function.*

Proof. We prove that $\mu_k[\cdot]$ is submodular by showing that it has decreasing marginal values. Let $\mathbf{y}^1, \mathbf{y}^2 \in \mathbb{B}^n$ and $l \in N$ be such that $\mathbf{y}^1 \lneqq \mathbf{y}^2$ and $\mathbf{y}^2 + \mathbf{e}_l \in \mathbb{B}^n$, where $\mathbf{e}_l \in \mathbb{B}^n$ is a vector whose entries are equal to 0 except for the l^{th} entry which is equal to 1. The vectors $\mathbf{y}^1$ and $\mathbf{y}^2$ can be viewed as indicator vectors of two subsets $N_1, N_2 \subset N$. Given our assumptions, $N_1 \subset N_2$ and $l \notin N_2$. We must show that

$$\mu_k\left[\mathbf{y}^1 + \mathbf{e}_l\right] - \mu_k\left[\mathbf{y}^1\right] \geq \mu_k\left[\mathbf{y}^2 + \mathbf{e}_l\right] - \mu_k\left[\mathbf{y}^2\right]. \tag{13}$$

For a set $N_q \subseteq N$, we define $k_q = \min(k, |N_i|)$ and S_q to be the set of indices of the k_q surgeries with highest values of d_i° in N_q, sorted in non-increasing order. Clearly, $k_1 \leq k_2$. We write that $S_1 = \{i_1, i_2, \ldots, i_{k_1}\}$ and $S_2 = \{j_1, j_2, \ldots, j_{k_2}\}$ with $d_{i_1}^\circ \geq d_{i_2}^\circ \geq \ldots \geq d_{i_{k_1}}^\circ$ and $d_{j_1}^\circ \geq d_{j_2}^\circ \geq \ldots \geq d_{j_{k_2}}^\circ$. As $N_1 \subset N_2$, we have that $d_{i_{k_1}}^\circ \leq d_{j_{k_2}}^\circ$. Define $N_3 = N_1 \cup \{l\}$ and $N_4 = N_2 \cup \{l\}$. When $k_2 < k$, then $S_3 = S_1 \cup \{l\}$ and $S_4 = S_2 \cup \{l\}$. Thus, the value of both left-hand side and right-hand side of (13) is equal to d_l°. However, when $k \leq k_2$, we must consider the following cases:

Case 1. $d^\circ_l \leq d^\circ_{i_{k_1}}$: In this case, $S_4 = S_2$. Two subcases may occur:

Case 1a. $k_1 \geq k$: In this subcase, $S_3 = S_1$. Thus, the values of the left-hand side and right-hand side of (13) are both equal to 0.

Case 1b. $k_1 < k = k_2$: In this subcase, $S_3 = S_1 \cup \{l\}$. Thus, the value of the left-hand side of (13) is equal to d°_l whereas that of the right-hand side is equal to 0.

Case 2. $d^\circ_{i_{k_1}} < d^\circ_l \leq d^\circ_{j_{k_2}}$: In this case, $S_4 = S_2$. Two subcases may occur.

Case 2a. $k_1 \geq k$: In this subcase, the right-hand side of (13) is equal to zero. However, $S_3 = S_1 \cup \{l\} \backslash \{i_{k_1}\}$ and consequently, the left-hand side of (13) is given by $\mu_k \left[\mathbf{y}^1 + \mathbf{e}_l\right] - \mu_k \left[\mathbf{y}^1\right] = \sum_{p=1}^{k_1-1} d^\circ_{i_p} + d^\circ_l - \sum_{p=1}^{k_1} d^\circ_{i_p} = d^\circ_l - d^\circ_{i_{k_1}} > 0$.

Case 2b. $k_1 < k$: In this subcase, $S_3 = S_1 \cup \{l\}$. Thus, the value of the left-hand side of (13) is equal to d°_l whereas that of the right-hand side is equal to 0.

Case 3. $d^\circ_l > d^\circ_{j_{k_2}}$: In this case, $S_4 = S_2 \cup \{l\} \backslash \{j_{k_2}\}$. Two subcases may occur.

Case 3a. $k_1 \geq k$: In this subcase, $S_3 = S_1 \cup \{l\} \backslash \{i_{k_1}\}$. Thus, the left-hand side of (13) is given by $\mu_k \left[\mathbf{y}^1 + \mathbf{e}_l\right] - \mu_k \left[\mathbf{y}^1\right] = \sum_{p=1}^{k_1-1} d^\circ_{i_p} + d^\circ_l - \sum_{p=1}^{k_1} d^\circ_{i_p} = d^\circ_l - d^\circ_{i_{k_1}}$ whereas its right-hand side is $\mu_k \left[\mathbf{y}^2 + \mathbf{e}_l\right] - \mu_k \left[\mathbf{y}^2\right] = \sum_{p=1}^{k_2-1} d^\circ_{j_p} + d^\circ_l - \sum_{p=1}^{k_2} d^\circ_{j_p} = d^\circ_l - d^\circ_{j_{k_2}}$. As $d^\circ_{i_{k_1}} \leq d^\circ_{j_{k_2}}$, we conclude that $d^\circ_l - d^\circ_{i_{k_1}} \geq d^\circ_l - d^\circ_{j_{k_2}}$ which shows that (13) holds.

Case 3b. $k_1 < k$: In this subcase, $S_3 = S_1 \cup \{l\}$. Thus, the value of the left-hand side of (13) is equal to d°_l whereas that of the right-hand side is given by: $\mu_k \left[\mathbf{y}^2 + \mathbf{e}_l\right] - \mu_k \left[\mathbf{y}^2\right] = \sum_{p=1}^{k_2-1} d^\circ_{j_p} + d^\circ_l - \sum_{p=1}^{k_2} d^\circ_{j_p} = d^\circ_l - d^\circ_{j_{k_2}}$. As $d^\circ_l > d^\circ_l - d^\circ_{j_{k_2}}$, (13) holds.

□

Given $\mathbf{y}_j \in [0,1]^n$, the *Lovász Extension* (Lovász 1983) of submodular function $\mu_k[\cdot]$ at $\mathbf{y}_j$ is defined to be

$$\mu_k^L[\mathbf{y}_j] = \sum_{i \in N} \left(\mu_k \left[\left(\sum_{l=1}^{i} \mathbf{e}_{\pi(l)} \right) \right] - \mu_k \left[\left(\sum_{l=1}^{i-1} \mathbf{e}_{\pi(l)} \right) \right] \right) y_{\pi(i),j}, \tag{14}$$

where $\pi : \{1, 2, \ldots n\} \to \{1, 2, \ldots n\}$ is a permutation such that $y_{\pi(1),j} \geq y_{\pi(2),j} \geq \ldots \geq y_{\pi(n),j}$. We denote the *Lovász Extension* of $\mu_k[\cdot]$ over the hypercube $[0,1]^n$ as $\mu_k^L[\cdot]$. It is known that the Lovász Extension of a submodular function defined over the vertices of the hypercube $[0,1]^n$ is in fact the convex envelope of this function over the entire hypercube $[0,1]^n$; see for instance Lemma 3.2 and

Corollary 2.7 of Tawarmalani et al. (2013). In this case, it can be verified that for any permutation $\pi : \{1, 2, \dots, n\} \mapsto \{1, 2, \dots, n\}$ and for any $\mathbf{y}_j \in [0, 1]^n$,

$$\mu_k^L[\mathbf{y}_j] \geq \sum_{i \in N} \left(\mu_k \left[\left(\sum_{l=1}^{i} \mathbf{e}_{\pi(l)} \right) \right] - \mu_k \left[\left(\sum_{l=1}^{i-1} \mathbf{e}_{\pi(l)} \right) \right] \right) y_{\pi(i),j},$$

showing that this envelope can be described by $n!$ linear inequalities. It is also easy to verify that $\mu_k^L[\mathbf{y}_j] = \mu_k \left[\mathbf{y}_j\right]$ for all $\mathbf{y}_j \in \mathbb{B}^n$. Hence, (12) reduces to

$$z_{j,k} \geq \mu_k^L[\mathbf{y}_j], \quad \forall k \in L \backslash \{0\}, \forall j \in M. \tag{15}$$

Further, replacing constraints (10e) of `zF` with (15) yields a model that computes the value of $z_{j,k}$ exactly at integer points $\mathbf{y}_j$ and has a tighter LP relaxation for `zF`. We refer to this modified `zF` formulation as `zF+L` and (15) as `Lovász-ineq` in the remainder of the paper.

For a given $(\mathbf{y}_j^*, z_{j,k}^*) \in [0,1]^n \times \mathbb{R}_+$, Algorithm 2 gives a polynomial time algorithm for separating inequalities (15). Using this algorithm, we introduce these constraints to the `zF+L` in a *lazy* manner. Similar to (10e) in the `zF`, we do so using callbacks in the commercial MILP solver.

Algorithm 2 Separation algorithm for (15)

```
1: Input: A vector $(\mathbf{y}_j^*, z_{j,k}^*)$ where $\mathbf{y}_j^* \in [0,1]^n$ for $j \in M$, $k \in L \backslash \{0\}$.
2: Output: An inequality of the form (15) violated by $(\mathbf{y}_j^*, z_{j,k}^*)$, if one exists.
3: Obtain permutation $\pi$ of $\{1, 2, 3, \dots n\}$ such that $y_{\pi(1),j}^* \geq y_{\pi(2),j}^* \geq \dots y_{\pi(n),j}^*$.    ▷ Step 1
4: Determine $\mu_k^L[\mathbf{y}_j^*]$ using (14).    ▷ Step 2
5: if $z_{j,k}^* < \mu_k^L[\mathbf{y}_j^*]$ then    ▷ Step 3
6:     return $z_{j,k} \geq \sum_{i \in N} \left( \mu_k \left[ \left( \sum_{l=1}^{i} \mathbf{e}_{\pi(l)} \right) \right] - \mu_k \left[ \left( \sum_{l=1}^{i-1} \mathbf{e}_{\pi(l)} \right) \right] \right) y_{\pi(i),j}$.    ▷ Step 3a
7: else
8:     return No violated inequality (15) exists for $(\mathbf{y}_j^*, z_{j,k}^*)$ for the given $j$ and $k$.    ▷ Step 3b
9: end if
```

For a given OR j and k, and $(\mathbf{y}_j^*, z_{j,k}^*) \in [0,1]^n \times \mathbb{R}_+$, Step 1 of Algorithm 2 involves determining a permutation π of surgeries such that $y_{\pi(1),j}^* \geq y_{\pi(2),j}^* \geq \dots y_{\pi(n),j}^*$ through sorting. In Step 2, the value of the *Lovász Extension* at $\mathbf{y}_j^*$, $\mu_k^L[\mathbf{y}_j^*]$ is determined using (14). Next, in Step 3, two possibilities are considered. If $z_{j,k}^*$ is less than $\mu_k^L[\mathbf{y}_j^*]$, then in Step 3a, an inequality defining (15) is returned. Otherwise, a statement stating that no such inequality exists is returned in Step 3b.

3.2.2 Valid inequalities for `zF` and `zF+L`

In this section, we introduce valid inequalities for (10) that can be included in `zF` and `zF+L`. Although these inequalities would be naturally satisfied by at least one optimal solution when the exponentially many inequalities (10e) are imposed, our implementation does not generate all constraints (10e).

Proposition 7. *The collection of inequalities*

$$z_{j,k} \geq z_{j,k-1} \qquad \forall k \in L \setminus \{0,1\}, \forall j \in M \tag{16a}$$

$$z_{j,1} \geq z_{j,2} - z_{j,1} \qquad \forall j \in M \tag{16b}$$

$$z_{j,k} - z_{j,k-1} \geq z_{j,k+1} - z_{j,k} \quad \forall k \in L \setminus \{0,1,\tau\}, j \in M, \tag{16c}$$

is satisfied by at least one optimal solution of (10)*.*

Proof. Consider any optimal solution $(\mathbf{x}^*, \mathbf{y}^*, \boldsymbol{\pi}^*, \mathbf{z}^*, \beta^*)$ of (10) where $z^*_{j,k} = \mu_k\left[\mathbf{y}^*_j\right]$ for each $j \in M$ and each $k \in L\setminus\{0,1\}$. We show that this solution satisfies (16a)-(16c). Pick $j \in M$ and sort the elements of $N = \{i_1, \ldots, i_n\}$ such that $d^\circ_{i_1} y^*_{i_1,j} \geq d^\circ_{i_2} y^*_{i_2,j} \geq \ldots \geq d^\circ_{i_n} y^*_{i_n,j}$. First, consider (16a) for a given $k \in L\setminus\{0,1\}$. We write that

$$z^*_{j,k} = \mu_k\left[\mathbf{y}^*_j\right] = \max_{\substack{S\subseteq N:\\ |S|\leq k}} \sum_{i\in S} d^\circ_i y^*_{i,j} = \sum_{p=1}^{k} d^\circ_{i_p} y^*_{i_p,j} \geq \sum_{p=1}^{k-1} d^\circ_{i_p} y^*_{i_p,j} = \max_{\substack{S\subseteq N:\\ |S|\leq k-1}} \sum_{i\in S} d^\circ_i y^*_{i,j} = \mu_k\left[\mathbf{y}^*_j\right] = z^*_{j,k-1},$$

where the inequality is due to the fact that $d^\circ_{i_k} y^*_{i_k,j} \geq 0$. Second, consider (16b). We write that

$$z^*_{j,1} + z^*_{j,1} = \mu_1\left[\mathbf{y}^*_j\right] + \mu_1\left[\mathbf{y}^*_j\right] = d^\circ_{i_1} y^*_{i_1,j} + d^\circ_{i_1} y^*_{i_1,j} \geq d^\circ_{i_1} y^*_{i_1,j} + d^\circ_{i_2} y^*_{i_2,j} = \max_{\substack{S\subseteq N:\\ |S|\leq 2}} \sum_{i\in S} d^\circ_i y^*_{i,j} = \mu_2\left[\mathbf{y}^*_j\right] = z^*_{j,2},$$

where the inequality holds because $d^\circ_{i_1} y^*_{i_1,j} \geq d^\circ_{i_2} y^*_{i_2,j}$. Third, consider (16c) for a given $k \in L\setminus\{0,1,\tau\}$. We write that

$$\begin{aligned} z^*_{j,k} - z^*_{j,k-1} =\ & \mu_k\left[\mathbf{y}^*_j\right] - \mu_{k-1}\left[\mathbf{y}^*_j\right] = \max_{\substack{S\subseteq N:\\ |S|\leq k}} \sum_{i\in S} d^\circ_i y^*_{i,j} - \max_{\substack{S\subseteq N:\\ |S|\leq k-1}} \sum_{i\in S} d^\circ_i y^*_{i,j} = \sum_{p=1}^{k} d^\circ_{i_p} - \sum_{p=1}^{k-1} d^\circ_{i_p} = d^\circ_{i_k} \\ \geq\ & d^\circ_{i_{k+1}} = \sum_{p=1}^{k+1} d^\circ_{i_p} - \sum_{p=1}^{k} d^\circ_{i_p} = \max_{\substack{S\subseteq N:\\ |S|\leq k+1}} \sum_{i\in S} d^\circ_i y^*_{i,j} - \max_{\substack{S\subseteq N:\\ |S|\leq k}} \sum_{i\in S} d^\circ_i y^*_{i,j} = z^*_{j,k+1} - z^*_{j,k}, \end{aligned}$$

which proves the result. □

As $z_{j,k}$ in (10) denotes the sum of the highest k values of d°_i among the surgeries assigned to OR

j, (16a) ensures that the sum of the k highest values of d_i° is at least as large as the sum of the $k-1$ highest values of d_i° among the surgeries assigned to OR j. Additionally, following the definition of $z_{j,k}$, for any $k \in L\backslash\{0,1\}$, $z_{j,k} - z_{j,k-1}$ is equal to the k^{th} highest d_i° value among the surgeries assigned to OR j. Constraint (16b) ensures that among the surgeries assigned to OR j, the highest d_i° value is at least as large as the second highest d_i° value whereas Constraint (16c) ensures that the k^{th} highest d_i° value is at least as large as the $(k+1)^{\text{th}}$ highest d_i° value. We refer to these inequalities as `z-ineq` in the remainder of this paper.

4 Integrating `C&CG` constructs into formulations `vF`, `zF`, and `zF+L`

In this section, we show that the formulations described in Section 3 can be integrated with the formulation that forms the basis of the `C&CG` algorithm for *2SRSSP* proposed in (Neyshabouri and Berg 2017, Zeng and Zhao 2013) to obtain a new formulation that, in our experiments, yields improved computational results. We provide the derivation of this hybrid formulation starting from `zF+L` although analogous hybrid formulations can be easily derived from `vF` and `zF`. An interested reader will find more details about the `C&CG` algorithm in Appendix A.

The `C&CG` formulation utilizes the same variables $\mathbf{x}$, $\mathbf{y}$, and β used in the formulations presented earlier in this paper. In addition, it generates the collection $\mathcal{S} = \{S_1, S_2, \ldots, S_p\}$ of all subsets of N of cardinality τ, indexed by the set $P := \{1, \ldots, p\}$. Each element k in P, which we refer to as a *scenario*, corresponds to one possible way (S_k) of extending the length of exactly τ surgeries. In addition, the `C&CG` formulation introduces continuous variables o_j^k to describe the amount of overtime incurred in OR j when scenario k is realized. The `C&CG` reformulation of *2SRSSP*, whose relaxation forms the master problem of the `C&CG` algorithm, is then written as

$$\begin{aligned}
v^* := \min\ & \textstyle\sum_{j\in M} c^f x_j + c^\circ \beta && && (17a)\\
\text{s.t.}\ & \sum_{j\in M} y_{i,j} = 1 && \forall i \in N && (17b)\\
& y_{i,j} \le x_j && \forall i \in N,\ \forall j \in M && (17c)\\
& \beta \ge \sum_{j\in M} o_j^k && \forall k \in P && (17d)\\
& o_j^k \ge \sum_{i\in N} d_i^l y_{i,j} + \sum_{i\in S_k} d_i^\circ y_{i,j} - U x_j && \forall k \in P, \forall j \in M && (17e)\\
& o_j^k \ge 0 && \forall k \in P, \forall j \in M && (17f)
\end{aligned}$$

$$\beta \geq 0,\ x_j \in \{0,1\},\ y_{i,j} \in \{0,1\} \qquad \forall i \in N,\ \forall j \in M. \tag{17g}$$

Objective (17a) and constraints (17b), (17c), and (17g) are identical to those in (1). Constraints (17e) compute the overtime experienced in OR j under scenario k, where durations d_i° are increasing the right-hand side value only for those surgeries i that are assigned to OR j and belong to the set S_k of surgeries whose duration is extended in this scenario. Constraints (17d) capture that the worst total overtime incurred is larger than the total overtime experienced in any of the scenarios.

As $|P|$ can be large, the C&CG algorithm solves relaxations of (17) obtained by imposing (17e) for subsets $P_s \subseteq P$ that grow larger with the iteration count s. During iteration s, the master problem is solved for P_s to obtain an optimal solution $(\mathbf{x}^s, \mathbf{y}^s, \mathbf{o}^s, \beta^s)$. The value of $F(\mathbf{x}^s, \mathbf{y}^s)$ is then computed by finding an optimal solution $\mathbf{r}^s$ to (2). If $\beta^s \neq F(\mathbf{x}^s, \mathbf{y}^s)$, then the scenario S_q corresponding to $\mathbf{r}^s$ is added to P_s to define $P_{s+1} = P_s \cup \{q\}$.

As new variables o_j^k must be introduced in each iteration for the new scenario, new columns are added to the master problem along with the additional constraints (17d)-(17f) these variables belong to. This justify the name *Column & Constraint Generation* for this algorithm. The generation of new columns is necessary to ensure the exactness of C&CG. However, the iterative inclusion of new variables during the *branch & bound* algorithm is not supported by existing commercial solvers. As a result, the MILP that forms the master problem has to be solved from scratch at each iteration. This significantly deteriorates the computational performance of C&CG, especially for problem instances with larger numbers of surgeries.

Next, we describe a formulation that integrates constraints (17d)-(17f) of the C&CG formulation into zF+L. This formulation has the advantage of combining the strength of both formulations while yielding an exact model that can be solved within the single *branch & bound* tree produced by commercial solvers. Specifically, the hybrid formulation we propose is

$$v^* = \min \sum_{j \in M} c^f x_j + c^\circ \beta \tag{18a}$$

$$\text{s.t.}\ (1c) - (1h) \tag{18b}$$

$$(6b) - (6d), (6f)(6g) \tag{18c}$$

$$(10c), (10d), (10f), (15) \tag{18d}$$

$$\beta \geq \sum_{j \in M} o_j^k \qquad \forall k \in P' \tag{18e}$$

$$o_j^k \geq \sum_{i \in N} d_i^l y_{i,j} + \sum_{i \in S_k} d_i^\circ y_{i,j} - U x_j \quad \forall k \in P', \forall j \in M \tag{18f}$$

$$o_j^k \geq 0 \quad \forall k \in P', \forall j \in M, \tag{18g}$$

where $P' \subseteq P$. Unlike (17), (18) is an exact reformulation of *2SRSSP* (1) for any $P' \subseteq P$ as (18c)-(18d) already ensure that β is at least equal to the worst-case overtime when at most τ surgeries take their maximum time. This allows us to only include (18e)-(18g) for any subset $P' \subseteq P$. We refer to these inequalities as `ccg-ineq`.

We thus propose to a priori fix the number of scenarios, $\tilde{K} = |P'|$ that we will include in (18) and set $P' = \{1, 2, \dots, \tilde{K}\}$. The model is initialized with constraints $o_j^k \geq 0$ instead of (18f) for $k \in P'$. The exact scenarios S_k to include in this collection will be determined as the *branch & bound* algorithm proceeds. In particular, we will derive these scenarios from the first $\tilde{K}$ *cost improving*, *robust feasible* solutions found in the *branch & bound* tree obtained using the commercial solver. In particular, for each *cost improving*, *robust feasible* solution $(\mathbf{x}^k, \mathbf{y}^k)$ found in the *branch & bound* tree for $k \leq \tilde{K}$, we solve the *second-stage* problem (5) for $(\mathbf{x}^k, \mathbf{y}^k)$ to determine the set of surgeries S_k, *i.e.*, the set of τ surgeries that take their maximum possible value. Then, we add the constraint (18f) for S_k using the callback features of the commercial MILP solver. Doing so allows us to continue the *branch & bound* algorithm without having to restart it. Once we have exceeded the pre-specified maximum number of scenarios $\tilde{K}$, these type of constraints are not updated anymore.

5 Lower bound for *2SRSSP* (1)

The hybrid formulation described above help in the solution of *2SRSSP* because it often improves the lower bounds obtained using the LP relaxation. In this section, we describe another approach to obtain a lower bound on the optimal value of *2SRSSP* under a mild condition. This lower bound can lead to the faster termination of the *branch & bound* algorithms for the three formulations.

Proposition 8. *For a set S^* of τ surgeries with highest d_i° values in N, define*

$$\tilde{m} = \left\lceil \frac{\sum_{i \in N} d_i^l + \sum_{i \in S^*} d_i^\circ}{U} \right\rceil . \tag{19}$$

If $\frac{c^\circ}{c^f} \geq \frac{1}{U}$ and $m \leq \tilde{m} - 1$, then $\tilde{L} = c^f m + c^\circ \left\{ \sum_{i \in N} d_i^l + \sum_{i \in S^*} d_i^\circ - mU \right\}$ is a lower bound for the optimal objective function value of 2SRSSP*.*

Proof. Let $(\mathbf{x}^*, \mathbf{y}^*)$ be a *robust optimal* solution of *2SRSSP* with optimal value v^* such that $\sum_{j \in M} x_j^* =$

$m - k \leq m \leq \tilde{m} - 1$ where $k \geq 0$. Let $\bar{\mathbf{r}} \in \mathbb{B}^n$ be the indicator vector of S^*, *i.e.*, $\bar{r}_i = 1$ if $i \in S^*$ and 0, otherwise. We write

$$\begin{aligned}
v^* \geq & \sum_{j\in M} c^f x_j^* + c^\circ \sum_{j \in M} \max\left\{0, \sum_{i\in N}\left(d_i^l + d_i^\circ \bar{r}_i\right) y_{i,j}^* - U x_j^*\right\} \\
\geq & \sum_{j\in M} c^f x_j^* + c^\circ \max\left\{0, \sum_{j\in M}\left(\sum_{i\in N}\left(d_i^l + d_i^\circ \bar{r}_i\right) y_{i,j}^* - U x_j^*\right)\right\} \\
= & \sum_{j\in M} c^f x_j^* + c^\circ \max\left\{0, \sum_{i\in N}\left(d_i^l + d_i^\circ \bar{r}_i\right) \sum_{j\in M} y_{i,j}^* - U \sum_{j\in M} x_j^*\right\} \\
= & \; c^f (m-k) + c^\circ \max\left\{0, \sum_{i\in N} d_i^l + \sum_{i\in S^*} d_i^\circ - U(m-k)\right\} \\
= & \; c^f (m-k) + c^\circ \left\{\sum_{i\in N} d_i^l + \sum_{i\in S^*} d_i^\circ - U(m-k)\right\} \\
= & \; c^f m + c^\circ \left\{\sum_{i\in N} d_i^l + \sum_{i\in S^*} d_i^\circ - Um\right\} + \left\{-c^f + c^\circ U\right\} k \\
\geq & \; c^f m + c^\circ \left\{\sum_{i\in N} d_i^l + \sum_{i\in S^*} d_i^\circ - mU\right\}
\end{aligned} \tag{20}$$

where the first inequality is due to the fact that all surgeries in S^* taking their maximum possible time is a feasible solution to the *second-stage* problem for $(\mathbf{x}^*, \mathbf{y}^*)$, the second inequality is due to the fact that $\sum_{l=1}^{W} \max\{0, a_l\} \geq \max\{0, \sum_{l=1}^{W} a_l\}$ for all $a_1, \ldots, a_W \in \mathbb{R}$, the second equality holds because of (1c) and the assumption that $\sum_{j\in M} x_j^* = m - k$, the third equality holds because of (19) and the assumption that $\tilde{m} \geq m + 1$, and the last inequality is due to the assumptions that $\frac{c^\circ}{c^f} \geq \frac{1}{U}$ and $k \geq 0$. □

Quantity $\tilde{L}$ can be interpreted as the optimal objective function value of a modified *2SRSSP* with N surgeries and one OR with capacity of mU and with fixed opening cost of $c^f m$. In this modified *2SRSSP*, the worst case overtime is given by $\sum_{i\in N} d_i^l + \sum_{i\in S^*} d_i^\circ - mU$ where τ surgeries with highest d_i° values take their maximum duration.

6 Computational experiments

In this section, we denote by `vF-ccg` the version of `vF` that incorporates `ccg-ineq`. Similarly, `zF-ccg` and `zF+L-ccg` refer to `zF` and `zF+L` with `z-ineq` and `ccg-ineq`. When applied to these formulations, *branch & bound* is terminated when a feasible solution is found with a relative MILP Gap of ϵ.

Finally, whenever we discuss the `four algorithms`, we refer to *branch & bound* applied to the three formulations `vF-ccg`, `zF-ccg`, and `zF+L-ccg`, and to `C&CG`.

The computational experiments utilize historical orthopedic surgery data from Mayo Clinic, an academic quaternary medical center and health system that operates campuses in Rochester, MN, Jacksonville, FL, and Scottsdale, AZ. Based on the data set for 10 working days, orthopedic surgeries last an average 221 minutes with standard deviation of 156 minutes. Based on this data, we fix $n = 25$ surgeries and consider four distinct values of m, namely $\{11, 14, 16, 18\}$. In accordance with Bansal et al. (2021) and Neyshabouri and Berg (2017), we assume that d_i^l for each orthopedic surgery follows a log-normal distribution, with an average of 221 minutes and a standard deviation (SD) of 156 minutes. Moreover, we define d_i° for surgery i as αSD, where α is drawn uniformly at random between 0.5 and 1.5. We determine the values of d_i° in this manner to introduce heterogeneity among surgeries. This heterogeneity arises because surgery times are influenced by various factors beyond the specialty itself, such as the patient's medical history and the specific type of surgery being performed.

We set U equal to 480 minutes and $\tau = \lceil \xi n \rfloor$ where $\lceil a \rfloor$ represents the integer nearest to a. We consider three values of $\xi \in \{0.3, 0.5, 0.7\}$ in our experiment. Because only the relative values of c^f and c° matter in the objective, we set c^f to 1 and $\frac{c^\circ}{c^f} = \chi = c^\circ$. Following Denton et al. (2010), we consider the two values of χ, $\frac{1}{30} = 0.0333$ and $\frac{1}{120} = 0.0083$. As $30 < 120 < 480 = U$, these values satisfy the condition that $\frac{c^\circ}{c^f} \geq \frac{1}{U}$ in Proposition 8.

Based on the pilot experiment described in Appendix B, we set $\tilde{K} = 25$ and add `ccg-ineq` to the three formulations for the first 25 *cost improving*, *robust feasible* solutions found by the commercial MILP solver. Before initiating the commercial MILP solver for both `zF-ccg` and `zF+L-ccg`, we include all the constraints (10e) and `Lovász-ineq` required to solve the LP relaxation of `zF-ccg` and `zF+L-ccg`, respectively. Furthermore, to enhance the root node relaxation for both formulations, we also add these inequalities for all fractional solutions that violate them at the root node of the *branch & bound* tree. These inequalities are added as *lazy* constraints to the formulation through callbacks in the commercial MILP solver. We set a time limit of $3,600$ seconds for each instance and each formulation. We enforce the same time limit for `C&CG`. We fix ϵ to be 0.01, meaning that we terminate any of the algorithms whenever it finds a feasible solution with a relative MILP Gap of 1%. Additionally, since all instances with $m = \{11, 14, 16\}$ satisfy the condition $m \leq \tilde{m} - 1$ as stated in Proposition 8, we terminate the *branch & bound* for these instances once a feasible solution within 1% of $\tilde{L}$ is found. We run our experiments using an intel(R) Xeon(R) Gold 6132 CPU @ 2.6 GHz processor with 128 GB RAM, Python 3.9, and GUROBI 10.0.2.

Next, we compare in Section 6.1 the computational performance of the three formulations `vF-ccg`, `zF-ccg`, and `zF+L-ccg` and of `C&CG`, the only other exact algorithm in the literature that has been specifically applied to *2SRSSP*. In Section 6.2, we investigate the computational effectiveness of `z-ineq`, `v-ineq`, and `ccg-ineq`. Lastly, in Section 6.3, we explore the strength of the lower bound $\tilde{L}$.

6.1 Comparison of the computational performance

To compare the computational performance of `vF-ccg`, `zF-ccg`, `zF+L-ccg`, and of `C&CG` for different values of m, we set $\xi = 0.5$ and perform a full factorial design across the considered values of m and χ. For each case, we generate five random instances and summarize the computational results in Tables 1, 2, and 3. Furthermore, to compare the computational performance of these `four algorithms` across different values of ξ, we fix $m = 16$ and conduct a full factorial design over the possible values of χ and ξ. Again, we generate five random instances for each case and present the computational results in Tables 4, 5, and 6.

To streamline the presentation, each case is assigned a distinct number, which is recorded in the first column of each table. The second column of each table contains the parameter values chosen for the associated case. The third to sixth columns of Table 1 and Table 4 give the average (maximum) MILP Gap for the `four algorithms`. For an instance solved using algorithm p, we calculate the MILP Gap as $\frac{UB^p - LB^p}{UB^p} \times 100$, where UB^p and LB^p are the best upper and lower bounds found by algorithm p for that instance when the algorithm is terminated. Tables 2 and 5 provide the percentage of instances in each case for which the `four algorithms` find a solution with a 1% MILP Gap within 3600 seconds. If such a solution is found for an instance, we consider the instance to be solved to 1%-optimality. Tables 3 and 6 give the average (maximum) computational time in seconds for the `four algorithms`.

Case #	Case χ—m	%-Gap-`vF-ccg`	%-Gap-`zF-ccg`	%-Gap-`zF+L-ccg`	%-Gap-`C&CG`
1	0.0333—11	0.80(1.50)	0.59(1.43)	0.79(1.80)	0.96(2.06)
2	0.0333—14	7.65(16.96)	7.29(14.26)	6.82(12.30)	7.26(14.00)
3	0.0333—16	5.33(12.98)	2.66(8.06)	3.47(10.70)	7.79(13.79)
4	0.0333—18	1.82(6.06)	0.92(0.99)	0.90(0.99)	3.18(5.98)
5	0.0083—11	0.38(1.39)	0.44(1.66)	0.58(1.39)	0.87(1.47)
6	0.0083—14	4.87(9.76)	5.04(9.84)	4.94(9.43)	5.43(10.38)
7	0.0083—16	5.15(8.79)	4.44(8.03)	4.36(8.51)	6.09(11.22)
8	0.0083—18	4.95(9.40)	3.36(8.30)	4.15(8.65)	5.96(10.53)

Table 1: Comparison between the `four algorithms` based on MILP Gaps for $m = \{11, 14, 16, 18\}$ and $\xi = 0.5$

Case #	Case χ—m	%-Inst-`vF-ccg`	%-Inst-`zF-ccg`	%-Inst-`zF+L-ccg`	%-Inst-`C&CG`
1	0.0333—11	80	80	80	80
2	0.0333—14	40	40	40	40
3	0.0333—16	20	60	60	20
4	0.0333—18	80	100	100	20
5	0.0083—11	80	80	80	80
6	0.0083—14	40	40	40	40
7	0.0083—16	40	40	40	20
8	0.0083—18	20	60	40	0

Table 2: Comparison between the `four algorithms` based on the percentage of instances solved to 1% MILP Gap in 3600s for $m = \{11, 14, 16, 18\}$ and $\xi = 0.5$

Case #	Case χ—m	Time-`vF-ccg`	Time-`zF-ccg`	Time-`zF+L-ccg`	Time-`C&CG`
1	0.0333—11	724(3600)	723(3600)	733(3600)	724(3600)
2	0.0333—14	2168(3600)	2199(3600)	2178(3600)	2182(3600)
3	0.0333—16	2983(3600)	2254(3600)	2136(3600)	2881(3600)
4	0.0333—18	1126(3600)	883(1971)	721(1598)	2895(3600)
5	0.0083—11	726(3600)	722(3600)	725(3600)	723(3600)
6	0.0083—14	2168(3600)	2164(3600)	2184(3600)	2177(3600)
7	0.0083—16	2322(3600)	2323(3600)	2298(3600)	2288(3600)
8	0.0083—18	3041(3600)	2616(3600)	2962(3600)	3600(3600)

Table 3: Comparison between the `four algorithms` based on computational time (in seconds) for $m = \{11, 14, 16, 18\}$ and $\xi = 0.5$

Case #	Case χ—ξ	%-Gap-`vF-ccg`	%-Gap-`zF-ccg`	%-Gap-`zF+L-ccg`	%-Gap-`C&CG`
1	0.0333—0.3	0.63(3.16)	1.21(2.85)	0.83(0.99)	4.82(7.22)
2	0.0333—0.5	5.33(12.98)	2.66(8.06)	3.47(10.70)	7.79(13.79)
3	0.0333—0.7	7.63(13.51)	7.89(13.49)	7.78(15.31)	6.32(11.50)
4	0.0083—0.3	2.80(6.66)	2.34(5.43)	2.29(6.16)	6.07(10.38)
5	0.0083—0.5	5.15(8.79)	4.44(8.03)	4.36(8.51)	6.09(11.22)
6	0.0083—0.7	5.37(9.07)	5.20(8.67)	5.21(8.79)	4.03(6.87)

Table 4: Comparison between the `four algorithms` based on MILP Gaps for $\xi = \{0.3, 0.5, 0.7\}$ and $m = 16$

Case #	Case χ—ξ	%-Gap-`vF-ccg`	%-Gap-`zF-ccg`	%-Gap-`zF+L-ccg`	%-Gap-`C&CG`
1	0.0333—0.3	80	80	100	0
2	0.0333—0.5	20	60	60	20
3	0.0333—0.7	20	20	0	20
4	0.0083—0.3	40	40	60	0
5	0.0083—0.5	40	40	40	20
6	0.0083—0.7	20	0	20	20

Table 5: Comparison between the `four algorithms` based on the percentage of instances solved to 1% MILP Gap in 3600s for $\xi = \{0.3, 0.5, 0.7\}$ and $m = 16$

Case #	Case χ—ξ	Time-`vF-ccg`	Time-`zF-ccg`	Time-`zF+L-ccg`	Time-`C&CG`
1	0.0333—0.3	1390(3600)	1050(3600)	1148(3600)	3600(3600)
2	0.0333—0.5	2983(3600)	2254(3600)	2136(3600)	2881(3600)
3	0.0333—0.7	2888(3600)	2884(3600)	2885(3600)	2881(3600)
4	0.0083—0.3	2845(3600)	2363(3600)	2243(3600)	3600(3600)
5	0.0083—0.5	2322(3600)	2323(3600)	2298(3600)	2288(3600)
6	0.0083—0.7	2887(3600)	2884(3600)	2883(3600)	2880(3600)

Table 6: Comparison between the `four algorithms` based on computational time (in seconds) for $\xi = \{0.3, 0.5, 0.7\}$ and $m = 16$

As can be seen in Tables 1, 2, and 3, all algorithms perform well for instances involving 11 ORs. These problems are relatively easy to solve because they are highly constrained due to the low total OR time availability with respect to the total surgery time to be scheduled. On the contrary, for the problem instances with $m = 14$, 16, and 18, we observe differences in the performance of all `four algorithms`. These differences are more prominent for cases with $m = 16$ and $m = 18$. Specifically, for the case 0.0333|16, the average MILP Gaps of solutions produced by `zF-ccg` and `zF+L-ccg` are 65.8% and 55.4% lower, respectively, compared to those of solutions generated by `C&CG`. Moreover, `zF-ccg` and `zF+L-ccg` achieve 1%-optimal solutions for 60% of the instances, whereas `C&CG` accomplishes this for only 20% of the instances. The solutions from `zF-ccg` and `zF+L-ccg` are achieved with average computational times that are 21.7% and 25.85% lower than those of `C&CG`. For the same case, `vF-ccg` is also outperformed by both `zF-ccg` and `zF+L-ccg` in terms of average MILP Gaps, percentage of instances solved to 1%-optimality, and average computational times. For case 0.0083|16, all `four algorithms` exhibit similar average computational times while both `zF-ccg` and `zF+L-ccg` outperform `C&CG` and `vF-ccg` in terms of average MILP Gap.

In case 0.0333|18, both `zF+L-ccg` and `zF-ccg` achieve 1%-optimal solutions across all instances. In contrast, `vF-ccg` achieves such solutions for 80% of the instances, whereas `C&CG` only manages this for 20% of them. Notably, `zF+L-ccg` achieves these high-quality solutions with 35.9% and 75% less average computational time compared to `vF-ccg` and `C&CG`, respectively. Furthermore, in this specific case, `zF+L-ccg` exhibits approximately an 18% lower average computational time than `zF-ccg`. In the 0.0083|18 case, `zF-ccg` achieves the smallest average MILP Gaps and solves 60% of the instances to 1%-optimality. In contrast, `C&CG` fails to solve any of the instances to 1%-optimality. It obtains these solutions with average computational times that are 11.6%, 14%, and 27% lower compared to `zF+L-ccg`, `vF-ccg`, and `C&CG`, respectively.

When examining the variation across different values of ξ for $m = 16$, distinct trends emerge in

Tables 4, 5, and 6 for all cases. In instances where $\xi = 0.3$, `C&CG` is notably outperformed by `vF-ccg`, `zF-ccg`, and `zF+L-ccg` in average MILP Gap, percentage of instances solved to 1%-optimality, and average computational time. Overall, for this case, `zF+L-ccg` exhibits the most favorable performance. In the 0.0333|0.5 case, `vF-ccg` and `C&CG` are markedly outperformed by `zF-ccg` and `zF+L-ccg` across all three criteria mentioned above while `zF-ccg` and `zF+L-ccg` demonstrate similar performance in this case. In the 0.0083|0.5 case, `vF-ccg`, `zF-ccg`, and `zF+L-ccg` outperform `C&CG` across all three criteria. Lastly, in cases with $\xi = 0.7$, `C&CG` delivers the best overall performance.

To summarize, the computational performance of `C&CG` (as measured by MILP Gap, computation times, and the percentage of instances where the algorithm finds a solution with a 1% MILP Gap within 3600 seconds) is outperformed by at least one of `vF-ccg`, `zF-ccg`, or `zF+L-ccg` for instances with $m = 16, 18$ surgeries and $\xi = 0.5$. Additionally, when comparing across different values of ξ, `C&CG` is outperformed by at least one of `vF-ccg`, `zF-ccg`, or `zF+L-ccg` for instances with $\xi = 0.3, 0.5$ and $m = 16$.

6.2 Effectiveness of `z-ineq` and `ccg-ineq`

In this section, we study the computational advantages of using valid inequalities `z-ineq` and `ccg-ineq`. We do so by eliminating each of these inequalities individually from `vF-ccg` and `zF+L-ccg` and by comparing the lower bounds obtained from the resulting MILP formulation with those obtained from `vF-ccg` and `zF+L-ccg`. As `zF+L-ccg` includes all the inequalities that we propose to strengthen `zF`, we exclude `zF-ccg` from this section. We compare these formulations based on the lower bound obtained after 3600 seconds of computation (LB-Final) and the root node lower bound (LB-Root). For this analysis, we consider the same cases and instances considered in Section 6.1.

Tables 7 and 8 present a comparison of lower bounds obtained from `zF+L-ccg` without `ccg-ineq` or `z-ineq` and `zF+L-ccg` for different values of m (with $\xi = 0.5$) and for different values of ξ (with $m = 16$), respectively. Specifically, the third and fourth columns of these tables record the average (maximum) % improvement in LB-Final and LB-Root achieved by incorporating `ccg-ineq` into `zF+L-ccg`. For LB-Final, we calculate the %-improvement as $\frac{LB^{Z_f} - LB^F}{LB^F} \times 100$, where LB^{Z_f} and LB^F are the best lower bounds obtained from `zF+L-ccg` and `zF+L-ccg` without `ccg-ineq`, respectively after 3600s of computation. For LB-Root, the %-improvement is calculated as $\frac{LB^{Z_r} - LB^R}{LB^R} \times 100$, where LB^{Z_r} is the root-node lower bound obtained from `zF+L-ccg`, and LB^R is the root-node lower bound from `zF+L-ccg` without `ccg-ineq`. The fifth and sixth columns present similar %-improvements for `z-ineq`. Likewise, Tables 9 and 10 give %-increase in LB-Final and LB-Root due to the addition of

		ccg-ineq		z-ineq	
Case #	Case χ—m	LB-Final	LB-Root	LB-Final	LB-Root
1	0.0333—11	31.47 (48.21)	53.04 (67.87)	0.33 (2.21)	0.09 (13.18)
2	0.0333—14	12.09 (36.07)	102.90 (147.10)	-1.44 (1.75)	-3.37 (0.00)
3	0.0333—16	1.25 (6.45)	50.05 (119.70)	0.21 (0.76)	0.71 (3.47)
4	0.0333—18	0.49 (2.28)	34.91 (126.41)	-0.01 (0.1)	-0.09 (0.00)
5	0.0083—11	25.32 (35.56)	31.68 (38.70)	0.55 (2.14)	0.44 (11.32)
6	0.0083—14	12.85 (35.04)	50.03 (64.88)	0.21 (0.75)	0.00 (0.00)
7	0.0083—16	6.31 (24.94)	27.61 (46.76)	-0.02 (0.79)	-0.06 (0.00)
8	0.0083—18	0.83 (2.24)	21.68 (53.21)	-0.51 (0.01)	-0.06 (0.02)

Table 7: Average (maximum) %-increase in LB-Final and LB-Root obtained from `zF+L` due to the addition of `ccg-ineq` or `z-ineq` for $m = \{11, 14, 16, 18\}$ and $\xi = 0.5$

		ccg-ineq		z-ineq	
Case #	Case χ—m	LB-Final	LB-Root	LB-Final	LB-Root
1	0.0333—0.3	0.01 (0.31)	19.59 (50.66)	0.15 (0.36)	4.31(21.44)
2	0.0333—0.5	1.25 (6.45)	50.05 (119.70)	0.21 (0.76)	0.71 (3.47)
3	0.0333—0.7	20.42 (63.97)	77.70 (158.20)	2.40 (7.84)	5.59 (27.95)
4	0.0083—0.3	-0.14 (0.02)	10.98 (18.82)	0.11 (1.05)	-2.52 (0.17)
5	0.0083—0.5	6.31 (24.94)	27.61 (46.76)	-0.02 (0.79)	-0.06 (0.00)
6	0.0083—0.7	16.88 (52.56)	37.12 (66.52)	3.04 (13.16)	-6.39 (19.79)

Table 8: Average (maximum) %-increase in LB-Final and LB-Root obtained from `zF+L` due to the addition of `ccg-ineq` or `z-ineq` for $\xi = \{0.3, 0.5, 0.7\}$ and $m = 16$

`ccg-ineq` in `vF`.

Tables 7, 8, 9 and 10 illustrate that including `ccg-ineq` substantially improves lower bounds for `zF+L-ccg` and `vF-ccg` across almost all cases. The inclusion of `z-ineq` has the most significant impact on cases with $\xi = 0.7$. In particular, for the case 0.0333|0.7, we observe an average improvement of 2.40% and 5.59% in LB-Final and LB-Root, respectively, due to the inclusion of `z-ineq`. Moreover, there are instances in this case where the improvement in LB-Final and LB-Root is as high as 7.84% and 27.95%, respectively. For case 0.0083|0.7, we observe an average improvement of 3.04%, with a maximum improvement of 13.16%, in LB-Final due the inclusion of `z-ineq` in `zF+L-ccg`.

6.3 Strength of lower bound $\tilde{L}$

In this section, we assess the strength of the lower bound $\tilde{L}$ by comparing it with the best lower bounds obtained from `vF-ccg`, `zF-ccg`, and `zF+L-ccg`. Additionally, we analyze the percentage of instances in each case for which $\tilde{L}$ led to an early termination of the *branch & bound* for the three

Case #	Case χ—m	LB-Final	LB-Root
1	0.0333—11	39.76 (65.72)	68.79 (105.16)
2	0.0333—14	11.45 (39.00)	109.08 (147.14)
3	0.0333—16	2.84 (12.89)	67.23 (120.42)
4	0.0333—18	-1.58 (-0.56)	35.52 (126.74)
5	0.0083—11	32.98 (51.60)	45.79 (59.12)
6	0.0083—14	9.90 (29.62)	50.23 (64.90)
7	0.0083—16	5.60 (9.14)	32.75 (46.91)
8	0.0083—18	2.64 (8.92)	21.53 (53.30)

Table 9: Average (maximum) %-increase in LB-Final and LB-Root obtained from `vF-ccg` due to the addition of `ccg-ineq` for $m = \{11, 14, 16, 18\}$ and $\xi = 0.5$

Case #	Case χ—m	LB-Final	LB-Root
1	0.0333—0.3	-0.43 (0.00)	27.10 (50.99)
2	0.0333—0.5	2.84 (12.89)	67.23 (120.42)
3	0.0333—0.7	20.87 (34.22)	118.89 (160.34)
4	0.0083—0.3	-0.86 (0.00)	14.65 (28.21)
5	0.0083—0.5	9.90 (29.62)	50.23 (64.90)
6	0.0083—0.7	15.33 (26.49)	53.78 (67.06)

Table 10: Average (maximum) %-increase in LB-Final and LB-Root obtained from `vF-ccg` due to the addition of `ccg-ineq` for $\xi = \{0.3, 0.5, 0.7\}$ and $m = 16$

formulations. We assess the effectiveness of the lower bound $\tilde{L}$ across various combinations of values of m and ξ. We explore the impact of different m values while holding ξ at 0.5, conducting a full factorial design across $m = \{11, 14, 16\}$ and $\chi = \{0.0333, 0.0083\}$. These values of m are considered because all instances with these values satisfy the condition $m \leq \tilde{m} - 1$ as stated in Proposition 8. For each case, we consider the same instances as considered in Section 6.1 and summarize computational results in Table 11. Subsequently, we examine the influence of ξ on $\tilde{L}$ by fixing m at 14 and conducting another full factorial design over the considered values of χ and ξ. For each case, we use the same instances as in Section 6.1 and present the results in Table 12.

The third column of Tables 11 and 12 are formatted as $a_1|a_2|a_3$, where a_1 represents the percentage of instances where the lower bound $\tilde{L}$ is at least 0.5% higher than the best lower bound determined by `vF-ccg`, a_2 denotes the percentage of instances where $\tilde{L}$ is at least 0.5% lower than the best lower bound determined by `vF-ccg`, and a_3 indicates the percentage of instances where $\tilde{L}$ is within 0.5% of the highest lower bound determined by `vF-ccg`. Columns four and five of both tables state the same for `zF-ccg`and `zF+L-ccg`. The sixth, seventh, and eighth columns of both tables, for each case, state the percentage of instances solved using `vF-ccg` (%-`vF`), `zF-ccg` (%-`zF`), and `zF+L-ccg` (%-`zF+L`), respectively, that terminated early due to the lower bound $\tilde{L}$. Recall that we embed this lower bound within the MILP solver using callback such that whenever a feasible solution within 1% of this lower bound is found, the solver terminates.

Case #	Case χ—m	`vF-ccg`	`zF-ccg`	`zF+L-ccg`	%-`vF`	%-`zF`	%-`zF+L`
1	0.0333—11	0—0—100	40—0—60	60—0—40	0	20	40
2	0.0333—14	0—40—60	0—20—80	0—20—80	0	0	0
3	0.0333—16	0—100—0	0—80—20	20—80—0	0	0	20
4	0.0083—11	0—0—100	0—0—100	40—0—60	0	0	20
5	0.0083—14	0—40—60	20—20—60	20—20—60	0	0	20
6	0.0083—16	0—100—0	0—80—20	20—80—0	0	0	20

Table 11: Evaluation of the strength of lower bound $\tilde{L}$ for $m = \{11, 14, 16\}$ and $\xi = 0.5$

As can be seen in Table 11, the relative strength of $\tilde{L}$ is more pronounced among cases with $m = 11$. Specifically, for all instances in cases with $m = 11$, $\tilde{L}$ is at least 0.5% better or within 0.5% of the highest lower bounds found in each of the three formulations. Additionally, for the case 0.0333|11, $\tilde{L}$ enables early termination in 40% and 20% of the instances solved using `zF+L` and `zF`, respectively. When comparing across different values of ξ in Table 12, we observe that for all cases with $\xi = 0.7$, $\tilde{L}$ is at least 0.5% better or within 0.5% of the highest lower bounds found in each of the three formulations. Additionally, for the case 0.0083|0.7, $\tilde{L}$ enables early termination in 40% of

Case #	Case χ—ξ	`vF-ccg`	`zF-ccg`	`zF+L-ccg`	%-`vF`	%-`zF`	%-`zF+L`
1	0.0333—0.3	0—80—20	0—80—20	20—80—0	0	0	20
2	0.0333—0.5	0—40—60	0—20—80	0—20—80	0	0	0
3	0.0333—0.7	0—0—100	40—0—60	20—0—80	0	20	0
4	0.0083—0.3	0—80—20	20—80—0	0—80—20	0	20	0
5	0.0083—0.5	0—40—60	20—20—60	20—20—60	0	0	20
6	0.0083—0.7	0—0—100	40—0—60	40—0—60	0	40	0

Table 12: Evaluation of the strength of lower bound $\tilde{L}$ for $\xi = \{0.3, 0.5, 0.7\}$ and $m = 14$

the instances solved using `zF`. For cases with $\xi = 0.5$, $\tilde{L}$ surpasses the highest lower bounds found in each of the three formulations by at least 0.5%, or it is within 0.5% of these bounds, for at least 60% of the instances.

In summary, the relative strength of $\tilde{L}$ is more significant in cases with $m = 11$ and $\xi = 0.5$. Furthermore, for these cases, including $\tilde{L}$ in `zF-ccg` and `zF+L-ccg` is computationally advantageous. Similarly, for cases with $\xi = 0.7$ and $m = 14$, including $\tilde{L}$ in `zF-ccg` is computationally beneficial.

7 Conclusion

In this paper, we model the *second-stage* of *2SRSSP* as a longest path problem on a layered acyclic graph. As a result, the *second-stage* problem can be represented as an LP, which can be dualized and integrated back into the *first-stage* model. This integration yields two monolithic MILP formulations for *2SRSSP*. We present methods to enhance their computational performance. Our numerical experiments demonstrate the computational advantages of these MILP formulations over `C&CG`. Furthermore, we show the effectiveness of the proposed methods to improve the computational efficiency of the MILP formulations. Future research will focus on exploring the polyhedral structure of these MILP formulations to derive tighter reformulations and reduce solution times further.

References

Addis B, Carello G, Tànfani E (2014) A robust optimization approach for the operating room planning problem with uncertain surgery duration. *Proceedings of the international conference on health care systems engineering*, 175–189 (Springer).

Ahuja RK, Magnanti TL, Orlin JB (1988) *Network flows* (Cambridge, Mass.: Alfred P. Sloan School of Management, Massachusetts).

Ardestani-Jaafari A, Delage E (2021) Linearized robust counterparts of two-stage robust optimization problems with applications in operations management. *INFORMS Journal on Computing* 33(3):1138–1161.

Bansal A, Berg BP, Huang YL (2021) A value function-based approach for robust surgery planning. *Computers & Operations Research* 132:105313.

Bansal A, Richard JP, Berg BP, Huang YL (2024) A sequential follower refinement algorithm for robust surgery scheduling. *INFORMS Journal on Computing* 36(3):918–937.

Denton BT, Miller AJ, Balasubramanian HJ, Huschka TR (2010) Optimal allocation of surgery blocks to operating rooms under uncertainty. *Operations Research* 58(4):802–816.

Gupta D (2007) Surgical suites' operations management. *Production and Operations Management* 16(6):689–700.

Lovász L (1983) Submodular functions and convexity. *Mathematical Programming The State of the Art: Bonn 1982* 235–257.

Marques I, Captivo ME (2017) Different stakeholders' perspectives for a surgical case assignment problem: Deterministic and robust approaches. *European Journal of Operational Research* 261(1):260–278.

Neyshabouri S, Berg BP (2017) Two-stage robust optimization approach to elective surgery and downstream capacity planning. *European Journal of Operational Research* 260(1):21–40.

Rath S, Rajaram K, Mahajan A (2017) Integrated anesthesiologist and room scheduling for surgeries: Methodology and application. *Operations Research* 65(6):1460–1478.

Schouten AM, Flipse SM, van Nieuwenhuizen KE, Jansen FW, van der Eijk AC, van den Dobbelsteen JJ (2023) Operating room performance optimization metrics: a systematic review. *Journal of Medical Systems* 47(1):19.

Tawarmalani M, Richard JPP, Xiong C (2013) Explicit convex and concave envelopes through polyhedral subdivisions. *Mathematical Programming* 138(1):531–577.

Zeng B, Zhao L (2013) Solving two-stage robust optimization problems using a column-and-constraint generation method. *Operations Research Letters* 41(5):457–461.

Appendix A: C&CG for *2SRSSP*

The C&CG algorithm involves reformulating the follower problem (2) as

$$
\begin{aligned}
F(\mathbf{x},\mathbf{y}) = \max_{\mathbf{r}} \min_{\boldsymbol{o}} & \sum_{j\in M} c^{\circ} o_j && && \text{(21a)}\\
\text{s.t.} \quad & o_j \geq \sum_{i\in N} \left(d_i^l + r_i d_i^{\circ}\right) y_{i,j} - U x_j && \forall j \in M && \text{(21b)}\\
& \sum_{i\in N} r_i \leq \tau && && \text{(21c)}\\
& r_i \in \{0,1\} && \forall i \in N && \text{(21d)}\\
& o_j \geq 0 && \forall j \in M, && \text{(21e)}
\end{aligned}
$$

where o_j gives the total overtime of OR j; see (Neyshabouri and Berg 2017). Constraints (21b), (21e), and the minimization in (21a) ensure that $o_j = \sum_{i\in N}\left(d_i^l + r_i d_i^{\circ}\right) y_{ij} - U x_j \geq 0$ if there is overtime in OR j, and 0 otherwise. To replace the maximum over $\mathbf{r}$ in the definition of $F(\cdot,\cdot)$, all feasible solutions r^k to the follower problem are enumerated. For each solution r^k, a variable o_j^k is introduced. By incorporating this representation into (1), a new formulation is obtained where the number of variables and constraints depend on the number of solutions r^k. Model (21) is solved by iteratively generating its constraints and variables. In its $\bar{K}^{\text{th}}$ iteration, C&CG solves the following master problem, which we refer to as $(CCG^{\bar{K}})$,

$$
\begin{aligned}
LB^{\bar{K}} = \min & \sum_{j\in M} c^f x_j + c^{\circ}\beta && && \text{(22a)}\\
\text{s.t.} \quad & \text{(1c)}, \text{(1d)}, \text{(1e)}, \text{(1f)}, \text{(1g)} && && \text{(22b)}\\
& \beta \geq \sum_{j\in M} o_j^k && \forall k \in \{1,2,\ldots,\bar{K}\} && \text{(22c)}\\
& o_j^k \geq \sum_{i\in N}\left(d_i^l + d_i^{\circ} r_i^k\right) y_{i,j} - U x_j && \forall j \in M, \forall k \in \{1,2,\ldots,\bar{K}\} && \text{(22d)}\\
& x_j \in \{0,1\},\ y_{ij} \in \{0,1\},\ o_j^k \geq 0 && \forall i \in N, \forall j \in M, \forall k \in \{1,\ldots,\bar{K}\}. && \text{(22e)}
\end{aligned}
$$

After obtaining an optimal solution $(\mathbf{x}^{\bar{K}}, \mathbf{y}^{\bar{K}}, \mathbf{o}^{\bar{K}}, \beta^{\bar{K}})$ for $(CCG^{\bar{K}})$, the *second stage* problem, (2), is solved to determine whether $(\mathbf{x}^{\bar{K}}, \mathbf{y}^{\bar{K}})$ is a *robust optimal* solution or if there exists a new $r^{\bar{K}+1}$ that improves upon $(CCG^{\bar{K}})$.

Algorithm 3 gives the pseudo-code for C&CG. The algorithm begins with initialization, where $\bar{K}$ is set to 0, LB is set to negative infinity, and UB is set to positive infinity. In Step 1 of the $\bar{K}^{\text{th}}$ iteration, $(CCG^{\bar{K}})$ is solved to determine $(\mathbf{x}^{\bar{K}}, \mathbf{y}^{\bar{K}})$ and $LB^{\bar{K}}$. In Step 2, the Lower Bound (LB) is updated to be equal to $LB^{\bar{K}}$. Next, in Step 3, the *second stage* problem, (2) is solved for $(\mathbf{x}^{\bar{K}}, \mathbf{y}^{\bar{K}})$ to determine $\mathbf{r}^{\bar{K}+1}$. In Step 4, the Upper Bound (UB) is updated based on the robust feasible solution produced in Step 3. Moving on to Step 5, $(CCG^{\bar{K}+1})$ is formulated using $\mathbf{r}^{\bar{K}+1}$. Then, in Step 6, $\bar{K}$ is incremented by 1, and the algorithm proceeds to the next iteration if the stopping condition $\frac{UB-LB}{UB} \leq \epsilon$ is not met. Here, ϵ is a small positive number. Proposition 2 of Zeng and Zhao (2013) demonstrates that Algorithm 3 terminates and yields an ϵ-optimal solution to *2SRSSP* in a finite number of iterations.

Algorithm 3 `C&CG` algorithm for *2SRSSP*

```
1: Input: Instance of 2SRSSP.
2: Output: Robust-optimal solution to 2SRSSP.
3: Initialize $\bar{K} \leftarrow 0$, $LB \leftarrow -\infty$, $UB \leftarrow \infty$.                         ▷ Step 0
4: while $\frac{UB-LB}{UB} \leq \epsilon$ do
5:     Solve $(CCG^{\bar{K}})$ to determine $(\mathbf{x}^{\bar{K}}, \mathbf{y}^{\bar{K}})$.                 ▷ Step 1
6:     $LB \leftarrow LB^{\bar{K}}$.                                                  ▷ Step 2
7:     Solve the second stage problem, (2) for $(\mathbf{x}^{\bar{K}}, \mathbf{y}^{\bar{K}})$ to obtain $\mathbf{r}^{\bar{K}+1}$.   ▷ Step 3
8:     $UB \leftarrow \min\{UB, \sum_{j \in M} c^f x_j^{\bar{K}} + c^{\circ} F(\mathbf{x}^{\bar{K}}, \mathbf{y}^{\bar{K}})\}$.   ▷ Step 4
9:     Formulate $(CCG^{\bar{K}+1})$ adding variables and constraints associated with $\mathbf{r}^{\bar{K}+1}$.   ▷ Step 5
10:    $\bar{K} \leftarrow \bar{K} + 1$.                                              ▷ Step 6
11: end while
12: return $(\mathbf{x}^{\bar{K}}, \mathbf{y}^{\bar{K}})$.
```

Appendix B: Results of the pilot experiment

We conduct a pilot experiment to determine the value of parameter $\tilde{K}$, the maximum number of `ccg-ineq` to be included in the formulations. We aim to add `ccg-ineq` for the first $\tilde{K}$ *cost improving*, *robust feasible* solutions found in the *branch & bound* algorithm. Additionally, we also investigate the impact on the computational performance when `ccg-ineq` is applied to the first $\tilde{K}$ *robust feasible* solutions instead of the first $\tilde{K}$ *cost improving*, *robust feasible* solutions. We refer to the former approach as *Implementation-1* and the latter approach as *Implementation-2*. We consider three candidate values of $\tilde{K} = \{25, 50, 75\}$. We conduct pilot experiments for cases with values of $\chi \in \{0.0333, 0.0083\}$. For this pilot experiment, we set $m = 14$ and $\xi = 0.5$, and use the `zF+L` formulation.

We run a full factorial design over the considered values of $\tilde{K}$ and χ, and solve three random instances for each case for a maximum of 3600 seconds.

Table 13 summarizes the result of the pilot experiment. The second column of the table specifies the cases considered. The third and fourth columns display the average (maximum) MILP Gap for *Implementations 1* and *2*, referred to in the table as %-Gap-1 and %-Gap-2, respectively. The fifth and sixth columns present, for each *implementation*, the average (maximum) number of `ccg-ineq` added during the *branch & bound*. These columns are labeled as #-Added-1 and #-Added-2, respectively. The last column (Nodes-5%) records, for each case, the fraction of instances in which the *branch & bound* for *Implementation-1* explored at least 5% more nodes than the *branch & bound* for *Implementation-2*.

Case #	Case χ—$\tilde{K}$	%-Gap-1	%-Gap-2	#-Added-1	#-Added-2	Nodes-5%
1	0.0333—25	12.13 (15.97)	12.40 (16.96)	18 (20)	25 (25)	66.67
2	0.0333—50	12.44 (17.24)	12.51 (16.79)	19 (23)	50 (50)	66.67
3	0.0333—75	12.03 (16.07)	13.91 (18.54)	25 (34)	75 (75)	66.67
4	0.0083—25	7.61 (10.32)	7.90 (11.09)	18 (20)	25 (25)	66.67
5	0.0083—50	7.82 (11.04)	8.04 (11.22)	20 (30)	50 (50)	100
6	0.0083—75	7.87 (10.81)	8.09 (11.59)	19 (28)	75 (75)	100

Table 13: Computational results of the pilot experiment

As observed in Table 13, *Implementation-1* yields a lower average MILP Gap across all cases when compared to *Implementation-2*. Additionally, *Implementation-1* yields a lower maximum MILP Gap across all cases but one (0.0333|50). This inferior performance can be attributed to the higher number of `ccg-ineq` inequalities added in *Implementation-2*, as evident from the fifth and sixth columns. The increased number of `ccg-ineq` inequalities in *Implementation-2* may result in higher solution time for the LPs in the *branch & bound*. Consequently, this results in a reduction in the number of nodes explored by the *branch & bound* during the allotted computational time. Specifically, *Implementation-2*, as compared to *Implementation-1*, explores at least 5% fewer nodes for at least two-thirds of the instances; see last column of Table 13. Exploring fewer nodes can lead to both weaker upper and lower bounds, which we believe is the reason behind the higher MILP Gap observed in *Implementation-2*. When comparing across the values of $\tilde{K}$ for *Implementation-1*, we find that $\tilde{K} = 25$ yields the lowest average (maximum) MILP Gap among all cases. Based on these results, in the main experiment, we add `ccg-ineq` for the first $\tilde{K} = 25$ *cost improving*, *robust feasible* solutions found by the commercial MILP solver.