Minimizing the maximum flow loss in the network maintenance scheduling problem with flexible arc outages

Shuang Jin[a], Ying Liu[b], Jing Zhou[a], Qian Hu[a,*]

[a]*School of Management and Engineering, Nanjing University, Nanjing 210093, China*
[b]*School of Economics and Management, Southwest Jiaotong University, Chengdu 610031, China*

Abstract

We investigate a network maintenance scheduling problem where maintenance tasks are carried out on the arcs of a network within flexible time windows. During maintenance, an arc is interrupted and no flow can pass through it. Such arc outages introduce flow loss and thus affect network capacity and service capability. For some public services operated on networks, possible blackouts and serious flow loss introducing extreme risks are generally unacceptable. The problem is to find a feasible schedule of maintenance tasks so that maximum flow loss during the planning horizon is minimized. We introduce a mixed integer programming formulation and a Benders reformulation for the problem. A Benders decomposition algorithm based on branch-and-cut is designed. Effective cuts are introduced to reduce the feasible region so that the exact algorithm is accelerated. Strengthened initial optimality cuts and an efficient separation procedure are also proposed. Computational experiments were conducted on a set of randomly generated instances from literature and a set of simulated instances based on telecommunication networks. Computational results show our algorithm performs much better than applying a solver to the formulation and an existing Benders decomposition algorithm for a related problem. The optimal schedules produced lower the extreme risks caused by large flow loss on the network. Since multiple optimal solutions may exist, hierarchical optimization is used to further select a desirable schedule, by either minimizing total flow loss or minimizing the duration of maximum flow loss. With adaptions, our algorithm also performs well for the two extensions.

Keywords: Scheduling, network maintenance scheduling, arc outage, flow loss, Benders decomposition

1. Introduction

With regular preventive maintenance, many failures and accidents can be prevented before they occur. Maintenance scheduling is to schedule a series of preventive maintenance tasks over a planning horizon. It is an effective way to improve service efficiency and reliability, reduce unexpected failure rate, and keep a

*Corresponding author.
Email addresses: shuangjin@smail.nju.edu.cn (Shuang Jin), liuyingsme@swjtu.edu.cn (Ying Liu), jzhou@nju.edu.cn (Jing Zhou), huqian@nju.edu.cn (Qian Hu)

system in good condition. Many services are provided through networks, such as power grids, water networks, and transportation networks. For such network-based services, maintenance tasks are operated on the networks and affect network capacity and service capability due to possible outages. Good scheduling solutions with well-planned maintenance tasks on the networks are critically important. Not only can preventive maintenance tasks reduce the risk of an accident that introduces network paralysis, but also a good maintenance schedule can reduce the adverse effects of outages on network capacity and service capability.

In network maintenance scheduling, some maintenance tasks are carried out on arcs, for instance, maintenance operations on rail tracks of a coal handing system (Boland et al., 2014), road maintenance and repair operations for urban transportation networks (Rey et al., 2019). Some tasks introduce arc outages, during which arc capacities become zero and no flow can pass through. Arc outages would reduce connectivity and cause *flow loss* on a network, i.e., the maximum flow is reduced. Boland et al. (2014) studied the network maintenance scheduling problem considering maximum total flow with flexible arc outages (MaxTFFAO). Arc outages are somewhat *flexible* because the start time of a task can be flexibly shifted within a time window. By optimizing the start times of all tasks to reduce flow loss, the MaxTFFAO aims to maximize the overall throughput of the network, i.e., the accumulated maximum flow over time. The objective meets the demand of some companies that seek to maximize the throughput of their production systems. For instance, a mining company tends to maximize the coal flow of a network from mines to ships (Boland et al., 2014).

However, for networks that provide public services, such as power grids, telecommunication networks, and transportation networks, only maximizing the total flow over time might be inappropriate. A major concern is that large-scale arc outages or blackouts are not acceptable. For instance, large-scale arc outages on a transportation network cause terrible traffic jams and possible accidents. Severe blackouts on the networks providing public services increase risks and dangers for people's life and could even introduce public panics. By analyzing some optimal solutions of the MaxTFFAO, for instance the example given in Section 3, we observe that for some time periods the network is shut down, without any flow on it, though a more robust schedule with less flow loss in the worst case exists. It thus motivates us to investigate how to schedule arc maintenance tasks with the aim of reducing large-scale flow loss or the worst flow loss for public services on the networks. Our new objective in the network maintenance scheduling problem is to minimize the maximum flow loss at any point in time.

In this work, we study a new variant of the MaxTFFAO, i.e., the network maintenance scheduling problem with the minimization of the maximum flow loss with flexible arc outages (MinMFLFAO). In the problem, a network and a set of arc maintenance tasks are given. Each task has a release date and a due date, i.e., it must be completed within a time window. If maintenance is carried out on an arc, no flow can pass through it for the duration of the maintenance activity. The problem is to find a feasible schedule determining the start times of the tasks so that the maximum flow loss during the horizon is minimized. The MaxTFFAO has been proved to be NP-hard by Boland et al. (2014) from reduction of a 3-partition problem

(Hartmanis, 1982). The MinMFLFAO involving a different objective is also NP-hard (see proof in Appendix A). Note that the objective of the MinMFLFAO identifies a bottleneck in network maintenance scheduling and multiple optimal solutions with the minimum maximum flow loss may exist. For this case, a secondary objective is helpful in finding a desirable schedule among these solutions. Hierarchical optimization is applicable for finding such a solution. In phase one, the MinMFLFAO is solved to obtain the minimum maximum flow loss. In phase two, the optimal value computed in phase one is imposed as an upper limit in constraints that restrict flow loss in each period. Two extensions, involving two different objectives, are considered for optimization in phase two. One minimizes the total flow loss over time and the other minimizes the total duration of maximum flow loss. To the best of our knowledge, there is no existing work on the MinMFLFAO and the extensions in the literature.

Our main focus is to develop a Benders decomposition algorithm for the MinMFLFAO. We first formulate the problem into a mixed integer programming (MIP) model by reducing min-max term in the objective. Following the work of Pearce & Forbes (2019), which introduced a disaggregated Benders decomposition algorithm for the MaxTFFAO, we reformulate our problem by applying Benders decomposition and develop a Benders decomposition algorithm that is implemented within a branch-and-cut framework. Particularly, we propose new design features including a set of no-good cuts that reduces the set of integer feasible solutions of the master problem, a set of goal cuts that reduces the feasible region of the linear programming (LP) relaxation of the master problem, a set of initial optimality cuts strengthened by tight time windows of maintenance tasks, and an efficient separation procedure of generating violated optimality cuts. They can effectively enhance our Benders decomposition algorithm. The algorithm is also adapted to solve the two problem extensions. Extensive computational experiments are conducted to evaluate the performance of the algorithm. Effectiveness of our algorithm and the new design features is verified with computational results. The results also suggest that our hierarchical optimization approach can find a desirable maintenance schedule among those with the minimum maximum flow loss.

The reminder of the paper is organized as follows. In Section 2, we provide a literature review on related problems, discuss new design features in our Benders decomposition algorithm, and summarize our contributions. In Section 3, the MinMFLFAO and its formulations as well as the two extensions are formally introduced. In Sections 4, we introduce Benders reformulation, suggest it is solved by a Benders decomposition algorithm within a branch-and-cut framework, and present two sets of cuts that facilitate solving the master problem. Section 5 presents the branch-and-cut implementation of the Benders decomposition algorithm and provides algorithmic enhancements including initial cuts and separation strategy. In Section 6, computational results on evaluating the performance of the algorithm and analyses on different problems are reported. We summarize our work in Section 7.

2. Related literature and key contributions

We first review the literature on network maintenance scheduling problems where different objectives are considered for different applications. Next, on network maintenance scheduling problems with flexible arc outages, we review the methodology and discuss the similarities and differences between an existing Bender decomposition algorithm and ours. Finally, the key contributions of this work is summarized.

2.1. Network maintenance scheduling problems with different objectives

Maintenance is now an important business function in many industries, such as power plants (Canto, 2008; Rodríguez et al., 2021), power grids (Froger et al., 2016), and transportation networks (Rey et al., 2019). It has received much attention from scholars. By surveying about two hundred papers published in the last two decades, de Jonge & Scarf (2020) conducted a dedicated review on maintenance optimization. Some scholars studied maintenance optimization considering maintenance and restoration after damage. Some other scholars focused on preventive maintenance, which is an effective way to improve service efficiency and reliability, reduce unexpected failure rate, and diminish unaffordable loss. In the scope of preventive maintenance, network maintenance scheduling is comparatively a narrow topic but also important in many applications. Below we briefly review the relevant literature on network maintenance scheduling problems, where different objectives are considered for different applications.

For the production systems of individual companies, which seek to improve total throughput and revenue, the network maintenance scheduling problems are to maximize the accumulated throughput or minimize the total capacity reduction of the network over the planning horizon. Tawarmalani & Li (2011) studied a network maintenance problem that is to schedule maintenance activities, in capacity-constrained time periods, on network arcs so as to minimize overall flow loss. They introduced integer programming formulations for the problem and studied the properties of optimal maintenance schedules. Boland et al. (2014) studied the network maintenance scheduling problem in the production system of a mining company, i.e., the MaxTFFAO as introduced. The coals are transported from mines to ships through a network infrastructure consisting of load points, railway tracks, and terminal equipment. Regular preventive maintenance tasks are flexibly scheduled and performed on the arcs of the network and the maintenance operations incur arc outages. The scheduling problem is to schedule a set of arc maintenance tasks to maximize the total throughput of the network. Boland et al. (2015) further discussed time discretization and storage at nodes in network maintenance scheduling, studied upper and lower bounds, and proposed a MIP model for the continuous time case. Boland et al. (2016) extended the work by considering an additional constraint to limit the number of tasks per time period. Kalinowski et al. (2020) studied a network maintenance scheduling problem from an application of a coal rail network company in Australia. A maintenance zone consists of some arcs connecting some contiguous track locations on the network and a maintenance task requires different resources and could introduce capacity reduction on the network. The problem is to schedule a set

of maintenance windows so that the total weighted capacity reduction over the annual planning horizon is minimized. The authors introduced a MIP model and developed a matheursitic for the problem.

For public services on networks, possible arc outages, risks of failures, and breakdowns introduced by maintenance tasks are non-negligible. On transportation networks, maintenance tasks may cause traffic congestion. Rey et al. (2019) studied a network maintenance scheduling problem based on road maintenance projects in a transportation network. Maintenance operations on roads cause partial or total road closures for some time periods and thus affect the traffic and network performance. The authors introduced a bilevel MIP formulation considering congestion effects and designed a branch-and-price algorithm for the problem. For power grids, maintenance may cause power outages and increase risks. Crognier et al. (2021) pointed out that some maintenance tasks on a power grid require a planned outage and the risk implied by performing maintenance has to be taken into account in the maintenance scheduling. The authors introduced a method to compute financial risk values of outages in different future scenarios and solved an optimization problem of minimizing a certain quantile of the risk distribution to find a robust schedule.

Considering the possible extreme risks introduced by maintenance operations, there is a bias to focus on the worst-case scenario and a robust objective should be considered. As a variant of the MaxTFFAO (Boland et al., 2014), the MinMFLFAO in our work finds a robust schedule that minimizes the maximum flow loss over the planning horizon. The objective with a min-max term is a bottleneck objective. It is useful to achieve equalization and avoid extremes for some bottleneck problems (Lin et al., 1998; Mäkinen et al., 2017; Bianchessi et al., 2022; Michael et al., 2022). With the bottleneck objective, a number of optimal solutions of MinMFLFAO may exist. A secondary objective helps to determine a desirable solution among them. For the MinMFLFAO, we consider two different secondary objectives to make a selection.

2.2. *Existing algorithms for network maintenance scheduling problems with flexible arc outages*

Both the MaxTFFAO and the MinMFLFAO are network maintenance scheduling problems with flexible arc outages, but they consider different objectives for their different applications. For the MaxTFFAO, Boland et al. (2014) introduced an integer programming formulation for the problem and developed six heuristics based on local search. Three data sets of test instances were generated and used for the evaluation of the heuristics. Following their work, Pearce & Forbes (2019) proposed a disaggregated Benders decomposition algorithm (DBD) for the MaxTFFAO. They implemented the algorithm within a branch-and-cut framework and proved that the Benders cuts are Pareto-optimal. Several techniques were used to improve the performance of the algorithm. Initial Benders optimality cuts based on cut sets are introduced, i.e., one minimum cut set of the network. More cut sets are obtained by increasing the capacity of arcs in the network. The LP relaxation of the master problem is strengthened by adding Benders optimality cuts as a warm-start. A feasible solution that is constructed using the results of master problem and subproblems is suggested to the solver as a heuristic solution. A hash table is used to store the results of subproblems so that the number of times of calling to solve a subproblem is reduced.

For many optimization problems involving two levels of decisions, Benders decomposition can be applied. According to the survey paper by Costa (2005), Benders decomposition has been successfully applied to a variety of network design problems. All these works involve a master problem that solves a relaxed problem to obtain values for a subset of variables and an auxiliary problem (or subproblem) that optimizes the remaining variables, i.e., flows on a network. Like the MaxTFFAO, the MinMFLFAO also involves two levels of decisions, i.e., the start times of maintenance tasks and the flows on the resultant network, and thus Benders decomposition algorithm is also applicable. Following the work of Pearce & Forbes (2019), we also study a Benders decomposition algorithm (BDA) for the MinMFLFAO, but propose new design futures. Below we point out the similarities and differences of the two algorithms.

Our algorithm is designed on the basis of DBD (Pearce & Forbes, 2019) and uses some of its effective features. In Benders decomposition, we also reformulate the problem with Benders optimality cuts and disaggregate the subproblem by time dimension into multiple subproblems. In the implementation of our Benders decomposition algorithm, we also adopt a branch-and-cut framework that is executed by a MIP solver. Inspired by the initial cuts used in DBD, we derive a new set of initial cuts, but they are strengthened in a different way. A hash table of storing solutions of subproblems and the proof of Pareto-optimal Benders cuts are also taken into account.

Serveral new design features in our BDA are proposed for the first time. They are effective in improving algorithm performance. First, we introduce two sets of cuts for the master problem so that the branch-and-cut procedure is accelerated due to reduced feasible regions. The first set of cuts are no-good cuts defined on the set of integer variables representing whether an arc is open or closed. They are helpful in reducing unpromising schedules and hence a number of integer solutions are pruned. The second set of cuts are goal cuts which simply define an integer bound on the objective. They are effective in reducing the feasible region of an LP relaxation and improving the upper bound. Second, instead of varying arc capacities, we introduce new initial optimality cuts by analyzing the time windows of maintenance tasks. If the time window of a task is too tight, the task must be in process for some time periods and the network with an arc outage certainly exists. Such networks with inevitable arc outages introduce different minimum cut sets and hence strong initial optimality cuts are obtained. Third, we design an efficient separation procedure to generate violated optimality cuts in the algorithm. It is invoked at both leaf nodes and non-leaf nodes, and a rounding heuristic is applied for a fractional solution when necessary. Finally, it is worth mentioning that our algorithm can be easily adapted to the two extensions in phase two of hierarchical optimization.

2.3. Contributions

We study a new network maintenance scheduling problem MinMFLFAO with a bottleneck objective, which is meaningful for networks that are sensitive to extreme risks. Considering the case that there may be multiple optimal solutions for the bottleneck objective, we further propose two extensions with a secondary

objective that leads to the selection of a better schedule, which is valuable for practical scheduling. The focus of the work is on designing a Benders decomposition algorithm for both the problem and its two extensions. The contributions of the paper can be summarized as follows:

- We present MIP reformulations for the problem and the two extensions. The decisions in the formulations are two-fold: the start times of the tasks and the flows on the arcs. Once the start times are fixed, the arc outages are determined. For each time period, the network might be different considering the possible interruptions on different arcs.

- We derive Benders reformulations for the problem and the two extension. In a reformulation, the master problem is to optimize the start times of the tasks and the subproblem for each time period is the maximum flow problem. To solve a reformulation, we develop a Benders decomposition algorithm and implement it within a branch-and-cut framework. To enhance the branch-and-cut for the master problem, we propose two sets of cuts: a set of no-good cuts that forbids some combinations of start times for some tasks so that the set of integer feasible solutions can be reduced, and a set of goal cuts that introduces integer bounds on the objective so that the LP relaxation is strengthened.

- We implement the algorithm within a branch-and-cut framework with further algorithmic enhancements. A set of initial optimality cuts that are strengthened considering inevitable arc outages is proposed to improve the upper bound at the root node. A new separation procedure is provided to efficiently solve maximum flow problems and generate violated Pareto-optimal optimality cuts.

- Extensive computational experiments are conducted to evaluate the performance of the algorithm and the new design features. A total of 240 test instances of the MaxTFFAO, as well as 260 simulated instances based on realistic telecommunication networks are used. The results show that our new design features are effective in improving algorithm performance. Our Benders decomposition algorithm significantly outperforms the direct MIP computed by a solver and DBD in terms of solution quality and computational efficiency. Furthermore, with analysis on the solutions, it indicates that the worst flow loss is significantly reduced and the schedule is better without introducing large-scale arc outages. The optimal schedules of the two extensions are further improved in a given metric.

3. Problem, extensions and formulations

3.1. Problem definition

In the MinMFLFAO, a set of maintenance tasks K and a network graph $G = (V, A)$ are given, where V is the set of nodes and A is the set of arcs. The network graph consists of $n + 2$ nodes, i.e., $V = \{0, 1, 2, \ldots, n, n + 1\}$, where 0 and $n + 1$ represent a source node and a sink node, respectively. Each arc $(i, j) \in A$ has a capacity, denoted as u_{ij}. Each maintenance task is carried out on an arc. Let $K_{ij} \subseteq K$ be the

subset of tasks that need to be carried out on arc $(i, j) \in A$. The release time and the deadline of task $k \in K$ is r_k and d_k, respectively; that is, the task needs to be started at or after r_k and completed no later than d_k. The processing time or the duration of task $k \in K$ is p_k and no preemption is allowed. The execution of task $k \in K_{ij}$ leads to an outage on arc (i, j) and its capacity becomes zero for p_k time periods. Once the task is completed, the capacity of arc (i, j) is resumed to u_{ij}. The scheduling horizon is H, i.e., $H = \{1, 2, \ldots, T\}$, where T is the last time period. Without loss of generality, we assume that u_{ij}, r_k, d_k, p_k are non-negative integers. Following Boland et al. (2014), for any two tasks on the same arc, we assume there is no overlap between their scheduling time intervals. That is, for any two task $k_1, k_2 \in K_{ij}$, either $d_{k_1} < r_{k_2}$ or $d_{k_2} < r_{k_1}$ is satisfied. As pointed out by Boland et al. (2014), any arc violating the assumption can be replaced by a path with several arcs and these intersecting tasks are distributed among different arcs of the path.

Let z_0 be the maximum flow on the network without any arc outage and, for a given maintenance schedule, let z_t denote the maximum flow on the network at time t. The *flow loss* at time t is defined as $z_0 - z_t$ and the *maximum flow loss* over the scheduling horizon is calculated as $\max_{t \in H}(z_0 - z_t)$. The MinMFLFAO is to decide the start times of the tasks and find a feasible schedule that minimizes the maximum flow loss.

An example network is illustrated in Figure 1. It is a simple network with maximum flow of 5. The source node is s and s' represents the sink node. For each arc, a letter and a number indicate arc name and arc capacity, respectively. We consider two example instances, given $T = 10$ and their task lists in Tables 1 and 2. Each row defines a maintenance task with its index, arc, duration, release time, and deadline. Some feasible schedules for the two instances are shown in Figures 2 and 3. The horizontal axis represents the time periods and the vertical axis represents the arcs. The numbers below the horizontal axis indicate the flow loss for each time period. For each task, a shaded rectangle shows its start and completion time in the schedule, and a dashed line shows its flexible time window, defined by its release time and deadline. In each figure, the left schedule is an optimal solution of the MaxTFFAO, and the right schedule is an optimal solution of the MinMFLFAO. In example (a), the MaxTFFAO schedule has a total flow loss of 22, but the network is totally shutdown for four time periods, during which the flow loss is equal to the maximum flow of 5; the MinMFLFAO schedule has a total flow loss of 22 and the maximum flow loss is 3, without network downtime. In example (b), the MaxTFFAO schedule has a total flow loss of 19 with network shutdown for three time periods; meanwhile, the maximum flow loss of the MinMFLFAO schedule is 3 without network downtime, but it yields a total flow loss of 22. We can see that the MinMFLFAO schedule distributes the tasks more evenly, reducing the maximum flow loss but may increase the total flow loss.

Following Boland et al. (2014) and Pearce & Forbes (2019), we derive the formulation of the problem using three sets of decision variables. Let y_k^t be the binary variables that equal 1 if task $k \in K$ is started at time $t \in H$ and 0 otherwise. Let x_{ij}^t be the binary variables that equal 1 if arc $(i, j) \in A$ is interrupted due to maintenance at time $t \in H$ and 0 otherwise. Variables $f_{ij}^t \in \mathbb{R}_+$ represent the amount of flow on arc $(i, j) \in A$ at time $t \in H$. For task $k \in K$, the earliest start time and the latest start time are r_k and $d_k - p_k + 1$, respectively.

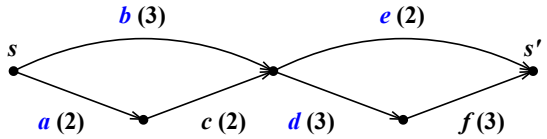


Figure 1: An example network with the maximum flow of 5

Table 1: A list of tasks in example (a)

k	arc	p_k	r_k	d_k
1	a	3	1	5
2	b	2	2	5
3	c	2	2	4
4	d	3	5	9
5	e	2	6	9
6	f	1	7	9

Table 2: A list of tasks in example (b)

k	arc	p_k	r_k	d_k
1	a	3	1	5
2	b	2	3	6
3	c	2	4	7
4	d	3	5	9
5	e	2	6	9
6	f	1	7	10

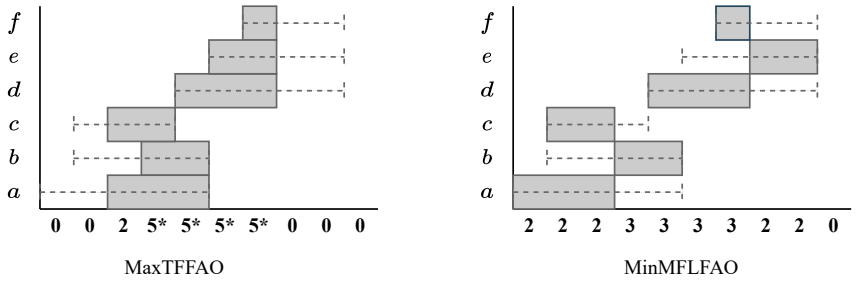


Figure 2: Two schedules of example (a)

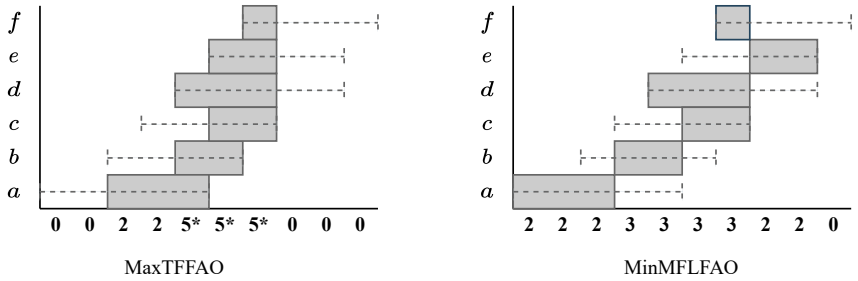


Figure 3: Two schedules of example (b)

The set of available start times for task k is denoted as H_k, i.e., $H_k = \{r_k, r_k + 1, \ldots, d_k - p_k + 1\}$. Thus, for any period $t \in H \setminus H_k$, we must have $y_k^t = 0$. Note that we must have $H_{k_1} \cap H_{k_2} = \emptyset$ for any two different tasks $k_1, k_2 \in K_{ij}$ on arc $(i, j) \in A$, following the assumption.

Table 3: Notations and symbols

Notation	Description
Sets and data	
$G = (V, A)$	a network graph, where V is the node set and A is the arc set
$V = \{0, 1, \ldots, n, n+1\}$	the index set of nodes, where 0 is the source node and $n+1$ is the sink node
u_{ij}	capacity of arc $(i, j) \in A$
K	the index set of arc maintenance tasks
K_{ij}	the subset of maintenance tasks on arc $(i, j) \in A$
p_k	duration (processing time) of task $k \in K$
r_k	release time of task $k \in K$
d_k	deadline of task $k \in K$
H	scheduling horizon, i.e., $H = \{1, 2, \ldots, T\}$, where T is the last time period
H_k	the set of candidate start times of task $k \in K$, i.e., $H_k = \{r_k, r_k + 1, \ldots, d_k - p_k + 1\}$
Decision variables	
$f_{ij}^t \in \mathbb{R}_+$	the flow on arc $(i, j) \in A$ at time $t \in H$
$z_t \in \mathbb{R}$	the maximum flow on the network at time $t \in H$
$y_k^t \in \{0, 1\}$	whether task $k \in K$ is started at time $t \in H$
$x_{ij}^t \in \{0, 1\}$	whether arc $(i, j) \in A$ is interrupted due to maintenance at time $t \in H$

With the notations and variables, which are summarized in Table 3, we formulate the MinMFLFAO as:

$$\text{(F1)} \quad \min \quad \max_{t \in H} \ (z_0 - z_t) \tag{1}$$

$$\text{subject to} \quad z_t = \sum_{(0,i) \in A} f_{0i}^t \qquad \forall t \in H \tag{2}$$

$$\sum_{(0,i) \in A} f_{0i}^t - \sum_{(i,n+1) \in A} f_{i,n+1}^t = 0 \qquad \forall t \in H \tag{3}$$

$$\sum_{(i,j) \in A} f_{ij}^t - \sum_{(j,i) \in A} f_{ji}^t = 0 \qquad \forall i \in V \setminus \{0, n+1\}, t \in H \tag{4}$$

$$f_{ij}^t \le u_{ij}(1 - x_{ij}^t) \qquad \forall (i, j) \in A, t \in H \tag{5}$$

$$x_{ij}^t = \sum_{k \in K_{ij}} \sum_{\tau == \max(1, t - p_j + 1)}^{t} y_k^\tau \qquad \forall (i, j) \in A, t \in H \tag{6}$$

$$\sum_{t \in H_k} y_k^t = 1 \qquad \forall k \in K \tag{7}$$

$$y_k^t = 0 \qquad \forall t \in H \setminus H_k, k \in K \tag{8}$$

$$y_k^t \in \{0, 1\} \qquad \forall k \in K, t \in H \tag{9}$$

$$x_{ij}^t \in \{0, 1\} \qquad \forall (i, j) \in A, t \in H \tag{10}$$

$$f_{ij}^t \ge 0 \qquad \forall (i, j) \in A, t \in H \tag{11}$$

In F1, objective (1) is to minimize the maximum flow loss in the planning horizon. Constraints (2) compute the maximum flow on the network at time $t \in H$ by summing up all the flows leaving the source node. Constraints (3) and (4) are the flow conversation constraints for each time period $t \in H$. Constraints (5) ensure that the flow on each arc cannot exceed the actual capacity; particularly, no flow can pass through it if an arc outage is incurred due to maintenance. Equations (6) show the relationship between variables $\mathbf{x}$ and $\mathbf{y}$. Constraints (7) and (8) guarantee that each task k can only be executed using one available start time from H_k. Constraints (9)-(11) define the domains of decision variables.

As z_0 is a constant which is presolved by any maximum flow algorithm, such as the Ford-Fulkerson algorithm (Ford & Fulkerson, 1956) or the Push-Relabel algorithm (Goldberg & Tarjan, 1986), to minimize the maximum flow loss $\max_{t\in H}(z_0 - z_t)$ in F1 is equivalent to minimizing the term $z_0 - \min_{t\in H} z_t$. Below, in F2, we introduce a new variable $\eta \in \mathbb{R}$ to denote the objective function of the problem, i.e., the minimum value of the maximum flow in the horizon. F2 is equivalent to F1, since η^* is the optimal value of F2 if and only if $z_0 - \eta^*$ is the optimal value of F1. We focus on solving F2.

$$
\begin{aligned}
\text{(F2)} \quad & \max \quad \eta \\
& \text{subject to} \quad \eta \le \sum_{(0,i)\in A} f_{0i}^t && \forall t \in H && (12) \\
& \qquad\qquad \text{Constraints (3)–(11)}
\end{aligned}
$$

3.2. Hierarchical optimization and two extensions

The objective in F2 characterizes a bottleneck of the MinMFLFAO. There may exist multiple alternative optimal solutions. Some of them are more preferable in terms of other metrics. A secondary objective is helpful in finding a desirable one. We thus apply hierarchical optimization with two phases. In phase one, the MinMFLFAO is solved through computing F2. In phase two, the following constraints are included:

$$\sum_{(0,i)\in A} f_{0i}^t \ge \eta^* \quad \forall t \in H \tag{13}$$

These ensure the flow loss for each time period does not exceed the optimal value of the first phase, η^*. The problem is also extended with a new objective function to decide the best schedule.

Two different secondary objectives are considered for selecting the best schedule. One minimizes the total flow loss over time and the other minimizes the total duration of maximum flow loss. Below, we introduce the two extensions and their formulations.

3.2.1. Minimize total flow loss in phase two

The first extension, denoted as E1, is to find a feasible schedule that minimizes the total flow loss over time and satisfies constraints (13). Note that it is equivalent to maximizing total flow over time, as expressed

by objective (14). We derive the following formulation for E1.

$$\text{(E1)} \quad \max \quad \sum_{t\in H}\sum_{(0,i)\in A} f_{0i}^{t} \tag{14}$$

$$\text{subject to} \quad \text{constraints (3)–(11), (13)}$$

Suppose $z(E1)$ is the optimal value of E1. The total flow loss in an optimal schedule must be $z_0T - z(E1)$.

3.2.2. Minimize total duration of maximum flow loss in phase two

The second extension, denoted as E2, is to find a feasible schedule that minimizes the total duration of maximum flow loss and satisfies constraints (13). To derive its formulation, we introduce binary decision variables ψ_t that equals 1 if the flow loss at time $t \in H$ achieves the maximum value $z_0 - \eta^*$ and 0 otherwise. The optimal binary values taken by ψ_t are determined by constraints (16) and objective (15). Due to the totally unimodular structure of the network flow problem, with given integer inputs, the flows are always integral. Binary variables ψ_t are safely relaxed to continuous as shown by constraints (17). Note that constraints (13) are not included as they are implied by constraints (16). The formulation E2 is given as follows.

$$\text{(E2)} \quad \min \quad \sum_{t\in H} \psi_t \tag{15}$$

$$\text{subject to} \quad \sum_{(0,i)\in A} f_{0i}^{t} + \psi_t \geq \eta^* + 1 \quad \forall t \in H \tag{16}$$

$$0 \leq \psi_t \leq 1 \quad \forall t \in H \tag{17}$$

$$\text{constraints (3)–(11)}$$

4. Benders reformulation and cuts

In F2, two levels of decisions are involved: the start times of tasks and the maximum flow of a network for each time period. Benders decomposition is naturally applicable for problems with such a structure. We present Benders reformulation for the MinMFLFAO (Section 4.1) and solve it using a Benders decomposition algorithm that is implemented within a branch-and-cut framework (Section 5). For the master problem, we introduce two sets of effective cuts (Section 4.2). One set of cuts reduces the set of integer feasible solutions by eliminating unpromising solutions and the other set of cuts imposes tight integer bounds on the objective so that LP relaxations are strengthened. For E1 and E2, which have similar problem structures, Benders reformulation and the algorithm are also applicable and we focus on the necessary adaptions.

4.1. Benders reformulation

In Benders decomposition of F2, it is naturally to move variables f_{ij}^t and constraints (3)–(5), (11) to subproblems, which are maximum flow problems, and retain other decisions variables and constraints in master problem. Let $\bar{\mathbf{x}}^t \subseteq \mathbb{B}^{|A|}$ denote a binary vector associated with variables x_{ij}^t at time $t \in H$. The vector decides a network with some arc outages. The maximum flow on the network at time t in the primal subproblem (PSP_t) is computed using the following LP formulation. Note that superscript t is removed from the variables representing arc flows for simplicity.

$$(PSP_t) \quad z_t(\bar{\mathbf{x}}^t) = \max \quad \sum_{(0,i)\in A} f_{0i} \tag{18}$$

$$\text{subject to} \quad \sum_{(0,i)\in A} f_{0i} - \sum_{(i,n+1)\in A} f_{i,n+1} = 0 \tag{19}$$

$$\sum_{(i,j)\in A} f_{ij} - \sum_{(j,i)\in A} f_{ji} = 0 \qquad \forall i \in V \setminus \{0, n+1\} \tag{20}$$

$$f_{ij} \le u_{ij}(1 - \bar{x}_{ij}^t) \qquad \forall (i,j) \in A \tag{21}$$

$$f_{ij} \ge 0 \qquad \forall (i,j) \in A \tag{22}$$

Let γ_0, γ_i, π_{ij} be the dual variables associated with constraints (19), (20), and (21), respectively. Additionally, we introduce an auxiliary variable γ_{n+1} and set its value by $\gamma_{n+1} = \gamma_0$. From duality in linear programming, the dual problem of PSP_t, denoted as DSP_t, has the following form:

$$(DSP_t) \quad z_t(\bar{\mathbf{x}}^t) = \min \quad \sum_{(i,j)\in A} u_{ij}(1 - \bar{x}_{ij}^t)\pi_{ij}$$

$$\text{subject to} \quad \gamma_0 - \gamma_i + \pi_{0i} \ge 1 \qquad \forall (0,i) \in A$$

$$\gamma_i - \gamma_j + \pi_{ij} \ge 0 \qquad \forall (i,j) \in A, i \ne 0$$

$$\pi_{ij} \ge 0 \qquad \forall (i,j) \in A$$

$$\gamma_i \in \mathbb{R} \qquad \forall i \in V$$

The feasible set of DSP_t is independent of t. Let Ω be the set of extreme points of the polyhedron of DSP_t. The optimal value of DSP_t is also the minimum solution value of all the extreme points, i.e.,

$$z_t(\bar{\mathbf{x}}^t) = \min_{(\boldsymbol{\pi},\boldsymbol{\gamma})\in\Omega} \sum_{(i,j)\in A} u_{ij}(1 - \bar{x}_{ij}^t)\pi_{ij}.$$

Thus, the Benders optimality cuts in the master problem are derived as

$$\eta \le \sum_{(i,j)\in A} u_{ij}(1 - x_{ij}^t)\pi_{ij}, \quad \forall (\boldsymbol{\pi},\boldsymbol{\gamma}) \in \Omega, t \in H.$$

Since the primal subproblem PSP_t always has a feasible solution given by $f_{ij} = 0$ for all $(i, j) \in A$, the Benders feasibility cuts do not need to be explicitly included into the master problem.

Finally, constraints (23) give the set of optimality cuts and the Benders reformulation of F2, denoted as MP, is given as follows.

$$\begin{aligned} \text{(MP)} \quad & \max \quad \eta \\ & \text{subject to} \quad \eta + \sum_{(i,j)\in A} u_{ij}\bar{\pi}_{ij}x^t_{ij} \le \sum_{(i,j)\in A} u_{ij}\bar{\pi}_{ij} \quad \forall(\bar{\boldsymbol{\pi}}, \bar{\boldsymbol{\gamma}}) \in \Omega, t \in H \\ & \qquad\qquad \text{constraints (6)–(10)} \end{aligned} \tag{23}$$

In the Benders decomposition algorithm for F2, the master problem is a restricted version of MP, where only a subset of optimality cuts characterized by $\bar{\Omega} \subseteq \Omega$ is provided. If any optimal solution of subproblem PSP_t or DSP_t violates constraints (12), i.e., $z_t(\bar{\mathbf{x}}^t) < \bar{\eta}$ for time t, a violated optimality cut is separated and added to the master problem.

The extensions E1 and E2, which have a similar problem structure, can also be reformulated via Benders decomposition. In Benders decomposition of E1, we do not consider to move constraints (13) to subproblems; otherwise, the subproblems are no longer maximum flow problems that are efficiently solvable, the master problem involves a number of feasibility cuts, and finally computational burden is significantly increased. Observing that the left-hand side of constraints (13) happens to be the optimal value of one maximum flow problem, it can be expressed using all the extreme points of dual subproblem DSP_t. We introduce a set of continuous variables η_t to represent the maximum flow of the network at time $t \in H$. Constraints (24) are used to reformulate constraints (13), and constraints (25) are Benders optimality cuts.

$$\sum_{(i,j)\in A} u_{ij}\bar{\pi}_{ij}x^t_{ij} \le \sum_{(i,j)\in A} u_{ij}\bar{\pi}_{ij} - \eta^* \quad \forall(\bar{\boldsymbol{\pi}}, \bar{\boldsymbol{\gamma}}) \in \Omega, t \in H \tag{24}$$

$$\eta_t + \sum_{(i,j)\in A} u_{ij}\bar{\pi}_{ij}x^t_{ij} \le \sum_{(i,j)\in A} u_{ij}\bar{\pi}_{ij} \quad \forall(\bar{\boldsymbol{\pi}}, \bar{\boldsymbol{\gamma}}) \in \Omega, t \in H \tag{25}$$

Moreover, it is convenient to impose bounds on variables η_t so that constraints (26) are also a substitute of constraints (13). As dominated by constraints (25) and (26), constraints (24) can be omitted.

$$\eta_t \ge \eta^* \quad \forall t \in H \tag{26}$$

Finally, he Benders reformulation of E1, denoted as E1-MP, has the following form:

$$\begin{aligned} \text{(E1-MP)} \quad & \max \quad \sum_{t\in H} \eta_t \\ & \text{subject to} \quad \text{Constraints (6)–(10), (25)–(26)} \end{aligned}$$

In the Benders decomposition algorithm for E1, the master problem is a restricted version of E1-MP, where only a subset of Benders cuts characterized by $\Omega_{E1} \subseteq \Omega$ is provided. If any optimal solution of subproblem PSP_t or DSP_t violates $z_t(\bar{\mathbf{x}}^t) \geq \eta_t$ for time t, a violated Benders cut is separated and added to the master problem.

Similarly, we can derive the Benders reformulation of E2, denote as E2-MP. Constraints (27) are a set of Benders cuts to reformulate constraints (16). The formulation is given as follows.

$$\begin{aligned} \text{(E2-MP)} \quad \min \quad & \sum_{t \in H} \psi_t \\ \text{subject to} \quad & \sum_{(i,j) \in A} u_{ij} \bar{\pi}_{ij} x_{ij}^t - \psi_t \leq \sum_{(i,j) \in A} u_{ij} \bar{\pi}_{ij} - \eta^* - 1 \quad \forall (\bar{\boldsymbol{\pi}}, \bar{\boldsymbol{\gamma}}) \in \Omega, t \in H \qquad (27) \\ & \text{Constraints (6)–(10), (17)} \end{aligned}$$

In the Benders decomposition algorithm for E2, the master problem is a restricted version of E2-MP, where only a subset of Benders cuts characterized by $\Omega_{E2} \subseteq \Omega$ is provided. If any optimal solution of subproblem PSP_t or DSP_t violates constraints (16), i.e., $z_t(\bar{\mathbf{x}}^t) < \eta^* + 1 - \bar{\psi}_t$ for time t, a violated Benders cut is separated and added to the master problem.

By exploiting the problem structure of F2, E1, and E2, the structure of subproblems in the reformulations are preserved, i.e., the classic maximum flow problem and minimum cut problems. As the subproblems are rapidly solved, our Benders decomposition algorithm in hierarchical optimization can run efficiently.

4.2. Cutting planes

Our Benders decomposition algorithm is implemented within a branch-and-cut framework that solves the master problem by branch-and-cut and separates violated optimality cuts at a node. Cutting planes play a key role in improving the performance of branch-and-cut. We introduce two sets of effective cuts for formulation MP. The first set includes no-good cuts defined on variables x_{ij}^t. They are effective in eliminating a number of unpromising solutions. The second set includes goal cuts which impose integer bounds on the objective and strengthen the LP relaxation.

4.2.1. No-good cuts

No-good cuts are often used to break the infeasibility or non-optimality by cutting the current solution (Hooker, 2000; Codato & Fischetti, 2006). We derive a set of no-good cuts that can break the current non-optimality so that a better feasible solution is expected to be found.

Suppose $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ is a feasible solution to constraints (6)–(10). With the set of available arcs determined by $\bar{\mathbf{x}}$, the maximum flow $\bar{z}_t(\bar{\mathbf{x}}^t)$ at time $t \in H$ is computed by solving either $PSP_t(\bar{\mathbf{x}}^t)$ or $DSP_t(\bar{\mathbf{x}}^t)$. Suppose t^* is a time period that the minimum maximum flow is obtained, i.e., $t^* = \arg\min_{t \in H} \bar{z}_t$. As $(\bar{\eta} = \bar{z}_{t^*}, \bar{\mathbf{x}}, \bar{\mathbf{y}})$ is a feasible solution to MP, we must have $\eta \geq \bar{z}_{t^*}$. It implies that, in an optimal solution, we have either

$\eta^* = \bar{z}_{t^*}$ or $\eta^* > \bar{z}_{t^*}$. For the first case, $(\bar{z}_{t^*}, \bar{\mathbf{x}}, \bar{\mathbf{y}})$ is an optimal solution and the algorithm is done. For the second case, all the solutions that use the same set of available arcs as defined by $\bar{\mathbf{x}}^{t^*}$ at any time period can be eliminated from the feasible set by adding to MP the following no-good cuts

$$\sum_{(i,j)\in A:\bar{x}^{t^*}_{ij}=0} x^t_{ij} + \sum_{(i,j)\in A:\bar{x}^{t^*}_{ij}=1} (1 - x^t_{ij}) \geq 1, \quad \forall t \in H.$$

The cuts enforce that at least one variable x^t_{ij} will take a different value from $\bar{x}^t_{ij}$.

For the maximum flow problem, more arc outages will never result in a better solution. Considering two sets of available arcs on the network, denoted as S_1 and S_2, we always have $z(S_2) \leq z(S_1)$ if $S_2 \subseteq S_1$, where $z(S)$ is the maximum flow on the network with the set of available arcs S. Let $D^* = \{(i,j) \in A : \bar{x}^{t^*}_{ij} = 1\}$ be the set of interrupted arcs determined by $\bar{\mathbf{x}}^{t^*}$. To achieve $\eta^* > \bar{z}_{t^*}$ in the second case, at least one interrupted arc from D^* must reopen. Therefore, the above cuts can be strengthened to a set of cuts, i.e.,

$$\sum_{(i,j)\in D^*} x^t_{ij} \leq |D^*| - 1, \quad \forall t \in H.$$

The set of cuts can be further improved considering the availability of an arc. Let W_k be the set of time periods in the time window of task k, i.e., $W_k = \{r_k, r_k + 1, \ldots, d_k\}$. For any time $t \in H \setminus \cup_{k\in K_{ij}} W_k$, no maintenance operation is proceeded on arc (i,j) and thus the arc is certainly open with $x^t_{ij} = 0$. This is also implied by constraints (6) and (8). Let $A^{open}_t = \{(i,j) \in A : t \in H \setminus \cup_{k\in K_{ij}} W_k\}$ be the set of arcs that must be open at time t. If an arc (i,j) from D^* also belongs to A^{open}_t, then we have $x^t_{ij} = 0$ and the no-good cut for time t always holds and thus is ineffective. We need to eliminate all such ineffective cases. Therefore, only the set of effective no-good cuts should be added to master problem, i.e.,

$$\sum_{(i,j)\in D^*} x^t_{ij} \leq |D^*| - 1, \quad \forall t \in \{\tau \in H : D^* \cap A^{open}_\tau = \emptyset\}.$$

Furthermore, for task k with a tight time window, i.e., $d_k - p_k < r_k + p_k$, an arc outage must occur from time $d_k - p_k$ to $r_k + p_k - 1$. Details are given in Section 5.2. Let A^{outage}_t be the set of arcs that must be interrupted at time t. With $D^*_t = D^* \setminus A^{outage}_t$, the no-good cuts can be further improved as

$$\sum_{(i,j)\in D^*_t} x^t_{ij} \leq |D^*_t| - 1, \quad \forall t \in \{\tau \in H : D^*_t \cap A^{open}_\tau = \emptyset\}.$$

Following this idea, it is possible to also derive no-good cuts for E1-MP and E2-MP. However, we do not use their cuts in the algorithm as they will not be as effective as those for MP. Because their objective functions accumulate a metric over time, the no-good cuts would involve a large number of variables x^t_{ij} with different time periods and thus can eliminate too few solutions.

4.2.2. Goal cuts

For an integer program, if the objective always takes integer values for any feasible solutions, the dual bound derived from LP relaxation can be strengthened considering integrality. For example, in a maximization case, a fractional upper bound can be rounded down. By imposing an integer upper bound on the objective, a goal cut can be used and is useful for our problem. It improves the LP relaxation of MP.

As integers are taken as inputs for capacities of all arcs, the optimal values of F2 and MP are always integral. By solving the LP relaxation of MP, we obtain an upper bound of the problem, denoted as UB. If UB is fractional, it can be rounded down to obtain a tighter upper bound, i.e., $\lfloor UB \rfloor$. We can impose the bound on the objective by adding to MP the following goal cut

$$\eta \leq \lfloor UB \rfloor.$$

In branch-and-cut, whenever the global upper bound UB is improved, the goal cut can be updated immediately. Therefore, it can be useful in reducing the feasible region of the LP relaxation at any node during the implicit enumeration of the branch-and-bound tree.

Furthermore, it also implies an optimality gap used in termination criterion. Let LB denote the global lower bound in the algorithm. The algorithm can be terminated earlier when $UB - LB < 1 - \epsilon$ is satisfied, where ϵ is a small constant denoting numeric precision, for instance, $\epsilon = 10^{-4}$ in our choice.

Similarly, another goal cut imposing a lower bound on the objective exists. Once a new feasible solution is found, the following cut can be added to the master problem,

$$\eta \geq LB + 1.$$

The cut helps determine whether a better feasible solution possibly exists for the newly explored node. If not, the node is pruned immediately. It is easy to apply goal cuts by just updating the bounds of variable η.

These cuts can be extended for E1-MP and E2-MP. For E1-MP, the goals cut are defined as $\sum_{t\in H} \eta_t \leq \lfloor UB \rfloor$ and $\sum_{t\in H} \eta_t \geq LB+1$. For the minimization formulation E2-MP, the goal cuts are defined as $\sum_{t\in H} \psi_t \geq \lceil LB \rceil$ and $\sum_{t\in H} \psi_t \leq UB - 1$. The new termination condition also applies to both formulations.

5. Benders decomposition algorithm

Following the work of Pearce & Forbes (2019), our Benders decomposition algorithm for the MinM-FLFAO is implemented within a branch-and-cut framework. In addition to the two sets of cuts, we present another two new design features for algorithmic enhancement, i.e., initial optimality cuts improved considering inevitable arc outages and an efficient separation procedure of generating violated optimality cuts.

5.1. Branch-and-cut implementation

Solving MP directly using a mathematical programming solver would be difficult. Because the formulation has an exponential number of constraints (the optimality cuts). A more efficient way to solve MP is to use a Benders decomposition algorithm. Rather than alternating between solving a master problem and subproblem, as in the classical Benders method, we solve a single master problem and generate Benders cuts on the fly. This approach was proposed by Hooker (2000), and later named "branch-and-check" by Thorsteinsson (2001) who successfully applied it to the problem of Jain & Grossmann (2001). It is actually a branch-and-cut routine and has a greater advantage when the master problem is difficult to solve.

The branch-and-cut implementation of our Benders decomposition algorithm relies on a MIP solver, i.e., Gurobi (Gurobi Optimization Inc, 2024). The master problem is solved once by the branch-and-cut routine of the solver and the violated cuts are separated at a node using its callback interfaces. The separation of violated Benders cuts is called at not only leaf nodes but also non-leaf nodes. The separation of violated no-good cuts is invoked at leaf nodes when an integer feasible solution is produced. Goal cuts are updated whenever a new global bound is obtained. The branching strategy, node selection, node pruning, and other operations in the branch-and-cut routine are controlled by the solver with default settings.

Algorithm 1 briefly show the framework of our algorithm. Let $\mathscr{T}$ denote the set of nodes to be explored. The algorithm starts to solve the root node with a restricted version of MP, denoted as RMP, where only a set of initial optimality cuts $\bar{\Omega}_t$ are included. It then iteratively solves the LP relaxations of RMP within the branch-and-cut framework and improves RMP by including the violated optimality cuts, which are separated by solving the dual subproblems. An LP rounding strategy is applied to $\bar{\mathbf{x}}^t$ for obtaining integer values for the separation of the optimality cuts. Let $\bar{\Omega}_t \subseteq \Omega$ denote the subset of optimality cuts for $t \in H$ in the current iteration of the algorithm, and $(\boldsymbol{\pi}^*, \boldsymbol{\gamma}^*)$ are cut parameters defining a set of optimality cuts. With optimality cuts generated during the early exploration of the search tree, RMP is strengthened for the subsequent nodes so that the upper bound can be possibly improved. Tight bounds are helpful in pruning nodes and further improving goal cuts. For no-good cuts, it examines whether an LP solution $(\bar{\eta}, \bar{\mathbf{x}}, \bar{\mathbf{y}})$ is integer feasible, and generates a set of no-good cuts for finding a better feasible solution. Let $\mathscr{D}$ be the set of no-good cuts added to RMP, and each cut is characterized by a pair (t, D_t^*), where t identifies the time period and D_t^* is the set of interrupted arcs. The RMP at a node is denoted by $RMP(\{\bar{\Omega}_t, D_t^*\}_{t\in H})$ and its LP relaxation is denoted by $RLMP(\{\bar{\Omega}_t, D_t^*\}_{t\in H})$.

5.2. Initial optimality cuts

Adding effective cuts at the root node can significantly improve the performance of a branch-and-cut algorithm. It improves dual bounds so that fewer nodes in the search tree will be explored. In our algorithm, a set of initial optimality cuts is added to RMP at the root node for the MinMFLFAO.

Algorithm 1: Benders decomposition algorithm for the MinMFLFAO

Data: A problem instance of the MinMFLFAO
Result: the best lower bound LB

$LB = 0$, $UB = +\infty$, $\mathscr{T} = \emptyset$, $\bar{\Omega}_t = \emptyset$, $D_t^* = \emptyset$, $\forall t \in H$;
Generate initial optimality cuts and include into $\bar{\Omega}_t$ for $t \in H$;
Create root node with $RMP(\{\bar{\Omega}_t, D_t^*\}_{t\in H})$ and add to set $\mathscr{T}$;
while $\mathscr{T}$ *is nonempty,* $UB - LB < 1 - \epsilon$ *and the time limit is not reached* **do**
    Select and remove a node from $\mathscr{T}$;
    Solve LP relaxation $RLMP(\{\bar{\Omega}_t, D_t^*\}_{t\in H})$ at the node;
    if *LP relaxation is feasible and node is not pruned* **then**
        Obtain an optimal LP solution $(\bar{\eta}, \bar{\mathbf{x}}, \bar{\mathbf{y}})$;
        for $t \in H$ **do**
            Apply rounding $\bar{x}_{ij}^t = \lfloor 0.5 + \bar{x}_{ij}^t \rfloor$ for $(i, j) \in A$ if $\bar{\mathbf{x}}^t$ is fractional;
            Solve $PSP_t(\bar{\mathbf{x}}^t)$ to obtain optimal value $z_t(\bar{\mathbf{x}}^t)$ and its residual graph;
            if $z_t(\bar{\mathbf{x}}^t) < \bar{\eta}$ **then**
                Compute cut parameters $(\boldsymbol{\pi}^*, \boldsymbol{\gamma}^*)$ from the residual graph;
                Include violated optimality cuts, i.e., $\bar{\Omega}_t = \bar{\Omega}_t \cup \{(\boldsymbol{\pi}^*, \boldsymbol{\gamma}^*)\}$;
            end
        end
        if *solution* $(\bar{\eta}, \bar{\mathbf{x}}, \bar{\mathbf{y}})$ *is integer feasible* **then**
            Update best incumbent solution and LB;
            Add the no-good cuts characterized by $(t, D_t^*) \in \mathscr{D}$ if $\bar{z}_t(\bar{\mathbf{x}}^t) = LB$;
        else
            Branch on the node and add child nodes to $\mathscr{T}$;
        end
        Update UB and goal cuts;
    end
end
return LB

Optimality cuts are determined by minimum cut sets, which are optimal solutions of dual subproblems. Let $(\boldsymbol{\pi}^*, \boldsymbol{\gamma}^*)$ be an optimal solution of DSP_t and C denote a minimum cut set. Particularly, π_{ij}^* takes a binary value indicating whether arc (i, j) is included in the minimum cut set. The minimum cut set is defined as $C = \{(i, j) \in A : \pi_{ij}^* = 1\}$. Let $\mathscr{C}$ include all the pairs (t, C) characterizing minimum cut set C of the network at time $t \in H$. Therefore, the optimality cuts (23) can be rewritten as

$$\eta \leq \sum_{(i,j)\in C} u_{ij}(1 - x_{ij}^t), \quad \forall (t, C) \in \mathscr{C}.$$

The cuts introduce an upper bound on the objective function η. Let C_0 represent a minimum cut set of network $G = (V, A)$, where no arcs are interrupted. A set of initial cuts represented by $\mathscr{C}_0 = \{(t, C_0) : t \in H\}$ is easily obtained.

With analysis on the time windows of maintenance tasks, we find a set of strong initial cuts exists. For a task with a tight time window, it is possible that the task must be in process for some particular time periods. As illustrated in Figure 4, tasks can be classified into two types according to their time windows. For the first type, a task k has a tight time window satisfying $d_k - p_k < r_k + p_k$ and the task must be processed from time $d_k - p_k$ to $r_k + p_k - 1$. For the second type, a task k has a loose time window with $d_k - p_k \geq r_k + p_k$; it is unclear for which time periods the associated arc is interrupted.

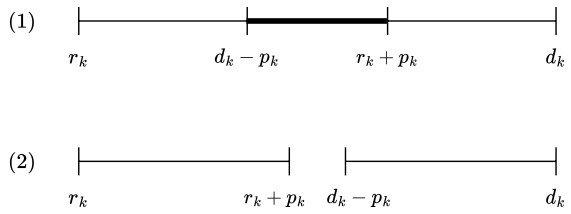


Figure 4: Two types of tasks with different time windows

To capture a bottleneck of the network at different time periods, all tasks of the first type are considered. Due to tight time windows, the associated arcs must be interrupted for some time periods. That is, some inevitable arc outages are observed. Let A_t^{outage} represent the set of arcs that must be interrupted at time t due to maintenance. For a network $G = (V, A \setminus A_t^{outage})$ at time t, a minimum cut set, denoted as C_t, is easily computed. As a result, a set of initial optimality cuts is obtained with $\mathscr{C}_1 = \{(t, C_t) : t \in H\}$. We include such strong initial optimality cuts to RMP at the root node.

For extension E1, we also obtain a set of initial cuts $\eta_t \leq \sum_{(i,j)\in C_t} u_{ij}(1 - x_{ij}^t)$ for $(t, C_t) \in \mathscr{C}_1$ and $t \in H$. For E2, initial optimality cuts are not applicable.

5.3. Separation procedure

An efficient procedure for generating optimality cuts is designed in our Benders decomposition algorithm. Given an optimal LP solution $(\bar{\eta}, \bar{\mathbf{x}}, \bar{\mathbf{y}})$ at a node, if $\bar{\mathbf{x}}^t$ is integral, the separation for a violated optimality cut is directly executed by solving the dual subproblem. In case $\bar{\mathbf{x}}^t$ is fractional, an LP rounding strategy is first applied to obtain an integer vector, i.e., $\bar{x}_{ij}^t = \lfloor 0.5 + \bar{x}_{ij}^t \rfloor$ for $(i, j) \in A$, and then the dual subproblem is solved. With the LP rounding strategy, the separation is also executed at non-leaf nodes so that more optimality cuts can be separated early for improving upper bounds.

A set of T networks are identified with $\bar{\mathbf{x}}^t$ for $t \in H$, each characterizes a primal subproblem $PSP_t(\bar{\mathbf{x}}^t)$ and a dual subproblem $DSP_t(\bar{\mathbf{x}}^t)$. One can solve the dual subproblem as linear programming and compute a dual optimal solution $(\boldsymbol{\pi}^*, \boldsymbol{\gamma}^*)$. Alternatively, we apply the classic Push-Relabel algorithm (Goldberg & Tarjan, 1986) to solve the maximum flow problem $PSP_t(\bar{\mathbf{x}}^t)$ in polynomial time. With the final residual graph computed in the algorithm, we can obtain a minimum cut set C for the minimum cut problem $DSP_t(\bar{\mathbf{x}}^t)$. Thus, an optimality cut characterized by $\boldsymbol{\pi}^*$ is generated, where $\pi_{ij}^* = 1$ for $(i, j) \in C$ and $\pi_{ij}^* = 0$ for

$(i, j) \in A \setminus C$. Suppose the optimal value of $PSP_t(\bar{\mathbf{x}}^t)$ is $z_t(\bar{\mathbf{x}}^t)$. Only if $z_t(\bar{\mathbf{x}}^t) < \bar{\eta}$, a minimum cut set C is computed and the optimality cut is added to *RMP*; otherwise, the cut is ineffective.

During the execution of the Benders decomposition algorithm, duplicate networks often arise. For instance, if a task is shifted a bit within its scheduling interval, the network structure remains the same for many time periods. Consequently, computing the maximum flow problem multiple times for the same network is computationally inefficient. To overcome this issue and enhance performance, we adopt the use of hash tables, as Pearce & Forbes (2019), to store the maximum flow and minimum cut of each network structure. To solve a subproblem, our algorithm first checks whether the network already exists in the hash table. If it does, the solution is retrieved immediately; otherwise, the Push-Relabel algorithm is utilized and the hash tables are updated accordingly. However, unlike Pearce & Forbes (2019), our approach involves storing the residual graph in the hash tables along with the maximum flow. Using the stored residual graph, a minimum cut set is fast computed only if it is necessary to generate the associated Benders cut.

In Benders decomposition, the dual subproblem could have multiple optimal solutions that do not yield cuts of equal strength. Magnanti & Wong (1981) selected a dual solution that dominates other possible cuts in terms of Pareto-optimality to find the strongest cuts. Like Pearce & Forbes (2019), the optimality cuts generated in our separation procedure are also Pareto-optimal, with proof in Appendix B. Utilizing these cuts can expedite the algorithm by reducing the number of cut separation rounds.

In the Benders decomposition algorithm for E1, the separation procedure can be applied directly for generating minimum cut sets and violated Benders cuts at both leaf and non-leaf nodes. As for E2, the separation for violated Benders cuts is executed at the nodes with an integer feasible solution obtained and the LP rounding strategy is not used.

6. Computational experiments

Extensive computational experiments were conducted to evaluate the performance of the proposed algorithm. It was implemented in Java and run single-threaded. If not particularly mentioned, the experiments were conducted on a workstation with an Intel(R) Xeon(R) Platinum 8372HC CPU x4 clocked at 3.40 GHz and 16 GB memory. The MIP solver used in the algorithm is Gurobi 11.0.0 (Gurobi Optimization Inc, 2024). All data sets and solutions are available at our Github page[1].

6.1. Instances

Boland et al. (2014) generated a set of random test instances based on eight different networks produced using the RMFGEN generater (Goldfarb & Grigoriadis, 1988) for the MaxTFFAO. These instances include all necessary data for our problem, so we use them to evaluate the algorithms for the MinMFLFAO as benchmark instances.

[1] https://github.com/njuora/minmflfao

The data sets from Boland et al. (2014) are divided into three classes, namely "dataset0", "dataset1", and "dataset2". In each of the three classes, there are eight sets of test instances, characterized by eight different networks from easy to difficult. The sizes of the networks are given in Table 4, where the columns *#node* and *#arc* represent the number of nodes and the number of arcs in a network, respectively. Each set includes 10 randomly generated test instances. Thus, there are 240 instances in total. In the generation of each instance, the planning horizon with $T = 1000$ is considered; the number of maintenance tasks on each arc ranges between 5 and 15; and the processing time of a task ranges between 10 and 30 time periods. But the scheduling interval of a task is different for the three classes. For instances in "dataset0", "dataset1" and "dataset2", the number of possible start times of a task is randomly generated using intervals [1, 10], [25, 35] and [1, 35], respectively. As the decision on starting a task is more flexible with more possible start times, the instances from "dataset1" are generally more difficult and the instances in "dataset0" are easier.

Table 4: Characteristics of networks

network	#node	#arc	network	#node	#arc
network1	12	32	network5	36	123
network2	16	44	network6	32	92
network3	18	57	network7	48	176
network4	27	90	network8	64	240

In addition, we generate simulated instances based on realistic telecommunication networks from SNDlib[2]. We add a source node s and a sink node s' to each network, and a set of arcs connecting to the two nodes are constructed. For a node without incoming arcs, an arc from node s to it is added. For a node without outgoing arcs, an arc from it to node s' is added. These newly added arcs have infinite capacity and no maintenance is required. In some instances, the pre-installed capacity of an arc is zero. In this case, we set the capacity of an arc to one unit of its modular capacity. All the nodes and arcs of these networks are shown in Table 5. In the maintenance plan of an instance, the planning horizon with $T = 365$ is used; the number of maintenance tasks ranges between 10 and 20 on each original arc; and task processing times vary between 3 and 7. For each network, ten distinct test instances are randomly generated with different time windows. There are 260 simulated instances in total.

6.2. *Analysis of the exact algorithm*

We first evaluate the key design innovations in the Benders decomposition algorithm (BDA) for the MinMFLFAO: no-good cuts, goal cuts, initial optimality cuts, and our separation procedure. In the comparison tests, BDA and its different implementations that disable a function were run on the eight test sets of "dataset1". For each run, a time limit of 3600 seconds was set. In the tables reporting comparison results,

[2]http://sndlib.zib.de

Table 5: Characteristics of realistic telecommunication networks

network	#node	#arc	network	#node	#arc
abilene	14	21	janos-us-ca	41	200
atlanta	17	30	newyork	18	55
brain	163	332	nobel-eu	30	53
cost266	39	79	nobel-germany	19	38
dfn-bwin	12	47	nobel-us	16	32
dfn-gwin	13	50	norway	29	54
di-yuan	13	44	pdh	13	36
france	27	57	pioro40	42	103
geant	24	47	polska	14	25
germany50	52	102	sun	29	156
giul39	41	250	ta1	26	61
india35	37	93	ta2	67	135
janos-us	28	136	zib54	56	107

the column *set* identifies a test set of 10 instances from a network, the columns *#opt* give the number of instances optimally solved for a test set, and the columns *#node* and *time (s)* give the number of explored nodes and the computational time in CPU seconds averaged on a test set, respectively. Numbers in bold indicate better results.

6.2.1. Analysis of no-good cuts

To examine the effectiveness of no-good cuts, an implementation of BDA without using any no-good cuts was run on the eight test sets. Table 6 reports the comparison results of BDA and the customized implementation. Note that all the instances are optimally solved by both implementations.

Table 6: Effectiveness of no-good cuts on the test sets of "dataset1"

set	BDA without no-good cuts			BDA		
	#opt	#node	time (s)	#opt	#node	time (s)
network1	10	1.7	2.00	10	1.0	**1.78**
network2	10	40.3	3.91	10	13.9	**3.22**
network3	10	15.1	6.95	10	2.5	**6.60**
network4	10	84.7	15.78	10	27.5	**11.50**
network5	10	58.5	21.80	10	4.6	**16.90**
network6	10	3.4	8.07	10	1.0	**7.48**
network7	10	635.6	129.58	10	366.3	**90.56**
network8	10	467.1	162.86	10	497.7	**138.90**
All	80	163.3	43.87	80	114.3	**34.62**

As observed from the table, for 7 out of 8 sets, the average number of explored nodes is reduced. All sets make an improvement in the average running time. No-good cuts are generally effective in speeding up BDA, with an average 30.0% reduction in the average number of explored nodes and an average 21.1% reduction in the average computational time.

6.2.2. *Analysis of goal cuts*

To verify the effectiveness of goal cuts, two customized implementations of BDA with disabling either the goal cut imposing an upper bound or the goal cut imposing a lower bound were run on the test sets, respectively. The comparison results of BDA and the customized implementations are reported in Table 7.

Table 7: Effectiveness of goal cuts on the test sets of "dataset1"

set	BDA without goal cut (ub)			BDA without goal cut (lb)			BDA		
	#opt	#node	time (s)	#opt	#node	time (s)	#opt	#node	time (s)
network1	10	16.9	2.68	10	1.0	2.00	10	1.0	**1.78**
network2	10	55.3	5.53	10	31.7	3.80	10	13.9	**3.22**
network3	10	256.9	13.86	10	15.2	6.98	10	2.5	**6.60**
network4	10	374.4	29.89	10	77.2	14.84	10	27.5	**11.50**
network5	10	397.8	42.16	10	23.0	19.68	10	4.6	**16.90**
network6	10	235.9	18.43	10	1.0	7.58	10	1.0	**7.48**
network7	10	1414.2	176.57	10	912.9	162.04	10	366.3	**90.56**
network8	10	1318.2	222.94	10	1075.4	248.97	10	497.7	**138.90**
All	80	508.7	64.01	80	267.2	58.24	80	114.3	**34.62**

As shown in the table, with the goal cut imposing an upper bound, the average number of nodes implicitly enumerated in the branch-and-bound tree by BDA is reduced by 77.5% and the average computational time is reduced by 45.9%. As for the goal cut imposing a lower bound, it reduces the average number of nodes by 57.2% and the average computational time is reduced by 40.6%.

6.2.3. *Analysis of initial optimality cuts*

Another customized implementation of BDA without using our initial optimality cuts $\mathscr{C}_1 = \{(t, C_t) : t \in H\}$ was also run on the test sets of "dataset1". The comparison results in Table 8 indicate the effectiveness of initial optimality cuts. The columns *#optimality_cut* give the number of optimality cuts added to RMP averaged on the 10 instances of a test set.

Table 8: Effectiveness of initial optimality cuts on the test sets of "dataset1"

set	BDA without initial optimality cuts				BDA			
	#opt	#optimality_cut	#node	time (s)	#opt	#optimality_cut	#node	time (s)
network1	10	3081.6	2.6	2.26	10	1295.4	1.0	**1.78**
network2	10	4362.6	5.8	3.24	10	2523.3	13.9	**3.22**
network3	10	5331.6	7.6	**6.24**	10	2826.2	2.5	6.60
network4	10	7004	18.7	11.07	10	4513.5	27.5	**11.50**
network5	10	8645.8	12.5	17.70	10	6538.3	4.6	**16.90**
network6	10	7009.5	4.8	7.90	10	4308.1	1.0	**7.48**
network7	10	32554.8	692.2	147.94	10	14419.7	366.3	**90.56**
network8	10	29593.4	547.5	185.88	10	18503.4	497.7	**138.90**
All	80	12197.9	161.5	47.78	80	6866.0	114.3	**34.62**

It indicates that our initial optimality cuts are effective in improving the performance of BDA. As observed from the table, the number of explored nodes is reduced for 6 out of 8 test sets, and the average

computational time is reduced on 7 out of 8 test sets. Initial optimality cuts are effective and able to improve the upper bound at root node. A tight bound is useful for pruning. As a result, the number of explored nodes is reduced, and the number of calls for the separation procedure is also reduced. On average, with initial optimality cuts, the number of explored nodes is reduced by 29.2% and the computational time is reduced by 27.5%. It further results a reduction in the number of optimality cuts, roughly 43.7%.

6.2.4. Analysis of separation procedure

The effectiveness of BDA also owes to our separation procedure, which is fast and applied at both leaf nodes and non-leaf nodes with LP rounding. Another two customized implementations were also executed on the test sets. The first one calls separation at only leaf nodes. In the second implementation, separation without LP rounding is applied at both leaf nodes and non-leaf nodes. Considering fractional values of variables $\bar{\mathbf{x}}$ in a master solution, $\bar{\mathbf{x}}$ also determine arc capacities in the network and the subproblems can also be computed for generating Benders cuts. The comparison results are reported in Table 9.

Table 9: Effectiveness of separation strategy on the test sets of "dataset1"

set	BDA with separation at only leaf nodes				BDA without LP rounding in separation				BDA			
	#opt	#optimality_cut	#node	time (s)	#opt	#optimality_cut	#node	time (s)	#opt	#optimality_cut	#node	time (s)
network1	10	468.8	2.9	2.12	10	884.8	1.0	2.03	10	1295.4	1.0	**1.78**
network2	10	836.3	63.3	4.57	10	1484.2	25.6	3.73	10	2523.3	13.9	**3.22**
network3	10	651.7	73.4	8.81	10	1199.9	31.5	7.34	10	2826.2	2.5	**6.60**
network4	10	1206.2	159.1	19.63	10	1994.8	42.6	13.88	10	4513.5	27.5	**11.50**
network5	10	1658.5	170.9	30.80	10	3141.4	9.3	18.66	10	6538.3	4.6	**16.90**
network6	10	1157.4	130.1	11.76	10	2414.3	2.3	8.59	10	4308.1	1.0	**7.48**
network7	10	2419.4	1091.9	146.56	10	3036.2	496.2	129.24	10	14419.7	366.3	**90.56**
network8	10	2119	1203.3	233.23	10	3092.8	352.6	187.26	10	18503.4	497.7	**138.90**
All	80	1314.7	361.9	57.18	80	2156.1	120.1	46.34	80	6866.0	114.3	**34.62**

Compared to BDA with separation at only leaf nodes, applying separation at non-leaf nodes can speed up the algorithm by reducing 66.8% explored nodes and 19.0% computational time on average. By further applying LP rounding in separation, BDA achieves a further reduction of 25.3% in computational time on average. It suggests that effective optimality cuts that are generated earlier in the search can better improve the upper bound and reduce nodes to be explored.

To examine the efficiency of our separation procedure, another two versions of BDA were tested: one solves the minimum cut problems directly using Gurobi, and the other also uses Gurobi in separation but no hash tables. Table 10 summarizes the comparison results. The columns t_{sep} *(s)* report the computational time on separating optimality cuts averaged on a test set. The results indicate that hash tables are effective and our separation method is faster than Gurobi. The average separation time saved by hash tables achieves 85.5%. With our separation procedure, the average separation time is further reduced by 44.6%.

In summary, all new design features are helpful in speeding up the algorithm. Particularly, initial optimality cuts and goal cuts are effective in improving the algorithm performance at an early stage. As we can

Table 10: Effectiveness of separation method on the test sets of "dataset1"

set	BDA without hash tables			BDA with separation by solver			BDA		
	#opt	t_{sep}(s)	time (s)	#opt	t_{sep}(s)	time (s)	#opt	t_{sep}(s)	time (s)
network1	10	1.99	3.76	10	0.40	2.10	10	0.04	**1.78**
network2	10	2.61	5.31	10	0.74	3.35	10	0.13	**3.22**
network3	10	5.37	11.24	10	1.14	6.81	10	0.23	**6.60**
network4	10	16.60	27.62	10	3.64	14.99	10	0.87	**11.50**
network5	10	13.78	27.04	10	4.82	18.07	10	2.00	**16.90**
network6	10	5.06	10.43	10	2.07	8.04	10	0.87	**7.48**
network7	10	162.54	257.29	10	21.68	99.96	10	11.92	**90.56**
network8	10	249.89	371.22	10	31.80	135.19	10	20.66	**138.90**
All	80	57.23	89.24	80	8.29	36.06	80	4.59	**34.62**

see, the number of nodes explored is largely reduced. It thus leaves a small improvement space for no-good cuts and separation strategy, which only works during the exploration of the branch-and-bound tree.

6.3. Comparison results

To evaluate the performance of our algorithm, we compared BDA to Gurobi that was applied to solve formulation F2, denoted as *F2(Gurobi)*, and DBD on all the benchmark instances. For each run of a method, a time limit of 3600 seconds was set.

We first reimplement the disaggregated Benders decomposition algorithm of Pearce & Forbes (2019), denoted as DBD, and apply it to the MaxTFFAO problem. The results of our implementation are compared to the results of DBD reported in Pearce & Forbes (2019) that are renamed as *DBD (PF19)*. Optimal solutions are quickly computed for all 80 instances of "dataset0" by both. We focus on the comparison results for "dataset1" and "dataset2" in Table 11. The columns *#opt* and *time (s)* give the number of instances optimally solved and the computational time averaged on a test set, respectively. It shows that our DBD implementation can solve 3 more instances to optimality and the average computational time is much less. It suggests the validity of our DBD implementation and provides a fair basis for subsequent comparisons.

Table 11: Comparison results of DBD between Pearce & Forbes (2019) and ours for the MaxTFFAO

class	set	DBD (PF19)		DBD (Our)		class	set	DBD (PF19)		DBD (Our)	
		#opt	time (s)	#opt	time (s)			#opt	time (s)	#opt	time (s)
dataset1	network1	10	57.4	10	36.89	dataset2	network1	10	21.3	10	11.19
	network2	**3**	3418.3	1	3455.06		network2	10	57.7	10	29.75
	network3	10	11.0	10	4.89		network3	10	8.3	10	5.41
	network4	2	2899.6	2	2895.19		network4	7	1540.1	7	1413.43
	network5	0	3600.7	**1**	3264.86		network5	0	3600.1	0	3600.18
	network6	0	3600.2	0	3600.18		network6	9	933.4	**10**	478.84
	network7	1	3314.2	1	3269.27		network7	3	2633	**4**	2449.69
	network8	7	1560.2	**8**	912.88		network8	8	927.3	**9**	595.30
	All	33	2307.70	33	2179.90		All	57	1215.15	**60**	1072.97

For the MinMFLFAO, Table 12 shows the comparison results for the hard benchmark instances from "dataset1", and the results for the other easy benchmark instances are provided in Appendix C. In the table,

the columns *set* and *class* identify a test set of 10 instances from one class. The columns *#opt* and η give the number of instances optimally solved and the average of best solution values in formulation F2 for a test set, respectively. The columns *#node* and *time (s)* report the number of explored nodes and the computational time in CPU seconds averaged on a test set, respectively.

Table 12: Comparison results on hard benchmark instances of the MinMFLFAO

class	set	*F2(Gurobi)*				DBD				BDA			
		#opt	η	#node	time (s)	#opt	η	#node	time (s)	#opt	η	#node	time (s)
dataset1	network1	10	29.8	246.8	9.94	10	29.8	132.5	3.75	10	29.8	1.0	1.78
	network2	10	31.5	507.0	31.08	10	31.5	172.7	5.94	10	31.5	13.9	3.22
	network3	10	91.2	2983.1	116.68	10	91.2	761.4	16.49	10	91.2	2.5	6.60
	network4	10	102.7	5808.3	400.30	10	102.6	666.7	35.75	10	102.7	27.5	11.50
	network5	10	83.2	2748.4	336.08	10	83.2	1331.6	56.42	10	83.2	4.6	16.90
	network6	10	26.9	1386.5	132.67	10	26.9	606.5	24.17	10	26.9	1.0	7.48
	network7	6	172.5	22795.6	2542.69	10	173.6	6665.6	294.45	10	173.6	366.3	90.56
	network8	6	159.6	10115.1	2299.27	9	160.7	30161.3	839.65	**10**	160.8	497.7	138.90
	All	72	87.2	5823.9	733.59	79	87.4	5062.3	159.58	**80**	87.5	114.3	34.62

As shown in Table 12, only BDA solved all of them optimally. F2(Gurobi) failed to solve 8 out of 80 instances to optimality. BDB failed to solve 1 out of 80 instances optimally. BDA outperforms the other methods in terms of both solution quality and computational efficiency. The improvement achieved by BDA is significant. Specifically, the average number of explored nodes are reduced by 98.0% and 97.7%, and the reductions in the average computational time are 95.3% and 78.3%, compared to Gurobi and DBD.

Due to the need for larger memory, for simulated instances, we switched to run the three methods on a workstation with an Intel(R) Xeon(R) CPU E5-2620 v4 clocked at 2.10 GHz and 64 GB memory. The comparison results are reported in Table 13, using similar columns as in Table 12. The columns *gap* (%) report the objective gap between upper bound and lower bound in percentage. It shows that BDA outperforms Gurobi and DBD significantly. On the 260 instances, Gurobi can only solve 110 instances to optimality, DBD can solve 132 instances optimally but BDA can optimally solve 207 instances. Only on sets "pioro40", "ta2" and "zib54", BDA has worse gap performance than Gurobi, which may compute better lower bounds using its heuristic routines; but such heuristics are not considered in BDA. Overall, compared to Gurobi and DBD, the average objective gap has been narrowed from 3.30% and 3.88% to 1.74% by BDA, the average number of explored nodes are reduced by 98.9% and 97.4%, and the reductions in the average computational time are 57.89% and 49.6%. In practice, faster optimization and better maintenance schedules are more beneficial in terms of cost savings and efficiency.

To better illustrate the comparison among the three methods, performance profiles are plotted in Figure 5. The abscissa indicates computational time, and the ordinates present the number and proportion of instances solved to optimality, respectively. As we can observe, the higher the curve on the upper left is, the better the performance of a method is. From the performance profiles, it is clear that BDA significantly

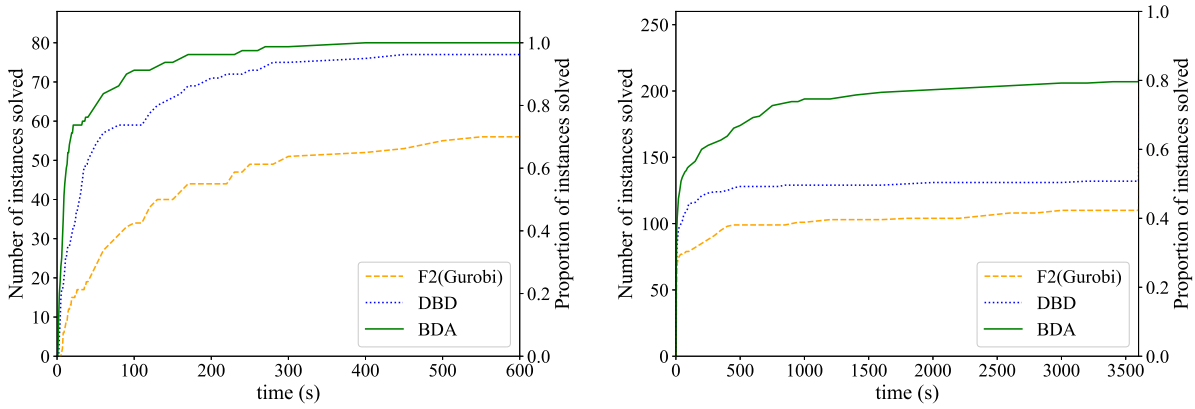
Table 13: Comparison results on simulated instances

network	*F2(Gurobi)*				DBD				BDA			
	#opt	gap (%)	#node	time (s)	#opt	gap (%)	#node	time (s)	#opt	gap (%)	#node	time (s)
abilene	10	0.00%	9854.7	92.31	10	0.00%	782.5	28.98	10	0.00%	623.8	39.42
atlanta	7	2.77%	111708.7	1288.74	9	1.56%	116817.8	759.74	**10**	0.00%	82.4	12.51
brain	10	0.00%	1.0	2.99	10	0.00%	1.0	2.04	10	0.00%	1.0	5.04
cost266	0	3.28%	6900.0	3600.11	0	4.91%	3648.0	3600.18	**6**	1.34%	1111.5	2720.43
dfn-bwin	5	4.27%	6501.6	2400.76	8	2.53%	1596.6	1032.36	**9**	0.00%	777.1	809.06
dfn-gwin	0	6.98%	61812.1	3600.58	1	5.24%	40566.9	3281.96	**10**	0.00%	375.5	116.92
di-yuan	6	3.86%	172154.7	2027.95	10	0.00%	15.4	8.32	10	0.00%	1.0	2.90
france	0	5.10%	23997.1	3600.32	0	7.68%	10630.6	3600.12	**7**	1.77%	981.1	1936.93
geant	0	5.28%	4369.2	3600.14	0	7.77%	3402.4	3600.10	**3**	2.29%	1137.6	2980.81
germany50	0	4.67%	1349.0	3600.36	0	14.72%	1049.6	3600.26	**6**	0.00%	396.5	1860.67
giul39	10	0.00%	1.0	7.49	10	0.00%	1.0	0.96	10	0.00%	1.0	2.10
india35	0	6.75%	4197.7	3600.29	0	7.71%	14877.5	3600.18	**8**	0.53%	721.7	1145.39
janos-us	10	0.00%	1.0	2.97	10	0.00%	1.0	0.42	10	0.00%	1.0	1.66
janos-us-ca	10	0.00%	1.0	4.84	10	0.00%	1.0	0.60	10	0.00%	1.0	0.98
newyork	0	8.46%	5425.1	3600.18	1	5.42%	16813.8	3281.77	**10**	0.00%	606.3	560.75
nobel-eu	0	4.92%	19769.0	3600.28	0	6.50%	25513.6	3600.07	**7**	2.41%	1362.6	1012.40
nobel-germany	4	2.51%	45263.7	2619.42	7	1.30%	7976.8	1369.35	**10**	0.00%	1.0	7.12
nobel-us	0	3.14%	54163.5	3600.22	1	2.90%	31925.7	3245.09	**10**	0.00%	108.6	48.93
norway	10	0.00%	24.8	4.29	10	0.00%	2.0	2.01	10	0.00%	1.0	2.44
pdh	10	0.00%	41154.8	343.97	10	0.00%	797.7	5.61	10	0.00%	1.0	2.19
pioro40	0	3.26%	1830.7	3600.29	0	8.63%	2087.3	3600.13	**1**	11.60%	321.8	3257.40
polska	5	7.21%	275953.7	2089.63	6	5.96%	101952.3	1492.01	**10**	0.00%	268.0	19.86
sun	10	0.00%	1.0	4.13	10	0.00%	1.0	0.60	10	0.00%	1.0	1.21
ta1	3	7.06%	23374.8	2585.89	9	1.41%	4242.2	418.95	**10**	0.00%	617.3	92.97
ta2	0	3.34%	823.2	3600.55	0	7.51%	873.1	3600.35	0	12.32%	228.2	3600.32
zib54	0	2.93%	1201.9	3600.28	0	9.02%	1128.4	3600.20	0	13.01%	236.6	3600.27
All	110	3.30%	33532.1	2179.96	132	3.88%	14873.3	1820.48	**207**	1.74%	383.3	916.95

outperforms the other two methods. Specifically, BDA compute all hard benchmark instances to optimality in less than 400 seconds, DBD can solve 96.25% of instances in 600 seconds, and Gurobi can not solve 70% instances in 600 seconds. For simulated instances, BDA solves many more instances to optimality than the other two methods within the same computational time.

The performance of our Benders decomposition algorithm that is adapted to the extensions E1 and E2 is also evaluated. Our algorithm is compared to Gurobi that is applied to the MIP formulations of E1 and E2, denoted as *E1(Gurobi)* and *E2(Gurobi)*, respectively. For each run, a time limit of 300 seconds was set. We provide comparison results on hard benchmark instances in Table 14 and 15, and the results on easy benchmark instances are in Appendix C. The new columns *#feasible* indicate the number of instances for which a feasible solution is found.

For E1, Gurobi can only solve 14 out of 80 instances optimally and find a feasible solution for 24 out of 80 instances within the time limit. Our BDA can optimally solve 30 instances and find a feasible solution for 61 instances. For E2, Gurobi can only solve 49 out of 80 instances optimally and find a feasible solution for 57 out of 80 instances within the time limit. Our BDA can optimally solve 73 instances and find a feasible solution for 76 instances. Overall, BDA performs significantly better than Gurobi for the two extensions.



(a) dataset1 (b) simulated instances

Figure 5: Performance profiles

Table 14: Comparison results of extension E1 on hard benchmark instances

class	set	*E1(Gurobi)*				BDA			
		#opt	#feasible	#node	time (s)	#opt	#feasible	#node	time (s)
dataset1	network1	4	10	6463.6	201.91	**10**	10	461.8	33.53
	network2	0	3	1362.4	300.05	**3**	**10**	1654.1	259.41
	network3	10	10	401.2	26.93	10	10	10.3	4.71
	network4	0	1	880.9	300.08	**3**	**9**	754.3	259.43
	network5	0	0	84.5	300.07	0	**2**	175.8	300.10
	network6	0	0	533.8	300.06	0	**9**	855	300.09
	network7	0	0	25.9	300.13	**1**	**5**	653.8	291.11
	network8	0	0	11.7	300.13	**3**	**6**	657.2	274.45
	All	14	24	1220.5	253.67	**30**	**61**	215.4	215.35

6.4. Analysis of different problems

Finally, we investigated the difference of schedules among the MinMFLFAO, the MaxTFFAO, and the extended problems E1 and E2. For the MaxTFFAO, we applied Gurobi to optimally solve its MIP formulation, which is similar to formulation F1 but utilizes a different objective to maximize the total maximum flow over the planning horizon, i.e., max $\sum_{t\in H} z_t$. For the MinMFLFAO, BDA was applied to obtain optimal schedules. For the two extensions E1 and E2, our adapted BDA was run to obtain optimal schedules. Particularly, we analyzed the flow loss, total flow loss, and duration of maximum flow loss of these optimal schedules.

6.4.1. Analysis of flow loss

First, we investigate different levels of flow loss between the MaxTFFAO and the MinMFLFAO schedules. The 10 instances in the test set from class "dataset1" using "network3" were considered. To further study how the results are affected by the flexibility of the scheduling time windows, based on the 10 instances, we considered to shorten and extend the time windows in the instances. Specifically, the deadline

Table 15: Comparison results of extension E2 on hard benchmark instances

class	set	*E2(Gurobi)*				BDA			
		#opt	#feasible	#node	time (s)	#opt	#feasible	#node	time (s)
dataset1	network1	10	10	258.5	12.14	10	10	27.8	2.22
	network2	9	10	1472.7	57.56	**10**	10	52.5	3.35
	network3	9	10	2599.8	117.01	**10**	10	261.4	9.82
	network4	6	9	2301.1	221.01	**10**	**10**	1080.2	30.81
	network5	6	7	1057.5	236.24	**10**	**10**	702.3	36.34
	network6	9	10	1066.7	131.41	**10**	10	277.6	15.62
	network7	0	0	730.1	300.15	**7**	**9**	4092	200.84
	network8	0	1	391.5	300.22	**6**	**7**	3553.5	228.33
	All	49	57	1234.7	171.97	**73**	**76**	1255.9	65.92

of a maintenance task k was set to $d_k + \Delta_{tw}$, where $\Delta_{tw} \in \{-5, 0, 5, 10\}$. When the time windows were modified, it was also ensured that all the tasks could be executed within the planning horizon, and that the time windows of the tasks on the same arc did not overlap. In total, 40 instances were used to compute the optimal MinMFLFAO schedules and the optimal MaxTFFAO schedules.

Table 16 shows the comparison results on flow loss for the two types of schedules. The flow loss in percentage at time t is computed as $100\% \times (z_0 - z_t)/z_0$, indicating the proportion of the maximum flow that is reduced at time t due to arc outages. Each row reports the results of a test set, i.e., 10 schedules for the 10 instances. The column *schedule* represents the type of the schedules. The column $\triangle_{tw}$ gives the change in the length of the scheduling time windows, a larger value indicates that the arc outages are more flexible. The columns under the block *flow loss (%)* report the total number of time periods during which the flow loss satisfies the condition in 10 schedules. For example, for the 10 test instances with $\triangle_{tw} = -5$, the MaxTFFAO schedules have a total of 8 time periods with flow loss exceeding 80% and 32 time periods with flow loss exceeding 70%.

Table 16: Flow loss analysis of two different types of schedules

schedule	Δ_{tw}	flow loss (%)						
		$\geq 80\%$	$\geq 70\%$	$\geq 60\%$	$\geq 50\%$	$\geq 40\%$	$\geq 30\%$	$\geq 20\%$
MaxTFFAO	-5	8	32	118	381	1022	2449	4178
	0	22	54	143	432	1175	2503	4202
	5	13	74	277	685	1285	2404	4071
	10	17	53	187	553	1208	2478	4262
MinMFLFAO	-5	0	0	0	0	0	1147	5655
	0	0	0	0	0	0	424	5636
	5	0	0	0	0	0	142	5519
	10	0	0	0	0	0	44	5526

The results show that the extreme risks due to maintenance can be reduced by solving the MinMFLFAO. There are no time periods with flow loss exceeding 40% in the MinMFLFAO schedules. In the MaxTFFAO schedules, the flow on the network fluctuates a lot during the planning horizon and severe flow loss exists for some time periods. The worst flow loss tends to be reduced if the arc outages are more flexible. As

we can see, by extending the time windows the number of time periods with flow loss reaching 30% is significantly reduced.

As for maximum flow loss, we show the comparison results in Figure 6. The abscissa indicates eight networks, each involves a group of 30 test instances from class "dataset0", "dataset1" and "dataset2" using the same network. A box plot represents the results of 30 instances. The maximum flow losses of the extended problems E1 and E2 are the same as that of the MinMFLFAO. We can see that the MaxTFFAO has more maximum flow loss than the MinMFLFAO, particularly on the mean and some quantiles.

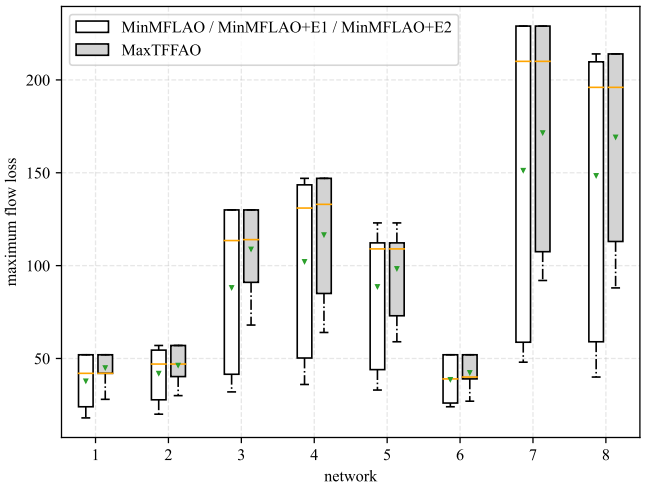


Figure 6: Comparison results on the maximum flow loss

6.4.2. *Analysis of total flow loss*

We compare the total flow loss for the MaxTFFAO, the MinMFLFAO, and the "MinMFLFAO+E1", as shown in Figure 7. In the firgure, each box represents the performance of 30 instances for a network in terms of total flow loss. It shows that the MinMFLFAO schedules achieve lower maximum flow loss but more total flow loss than the MaxTFFAO schedules. However, after E1 is solved in the second stage, the total flow loss of the obtained schedules reduce a lot, achieving almost the same total flow loss as the MaxTFFAO schedules. Therefore, our second phase in the hierarchical optimization is very effective and necessary. The "MinMFLAO+E1" schedules, which are obtained from the hierarchical optimization, successfully achieve the minimum maximum flow loss for avoiding extreme risks and acceptable total flow loss for preserving maximal network capacity and service capability.

6.4.3. *Analysis of duration of maximum flow loss*

The MaxTFFAO schedules, the MinMFLFAO schedules, and the "MinMFLFAO+E2" schedules are also compared by evaluating the duration of maximum flow loss, as shown in Figure 8. The "MinM-

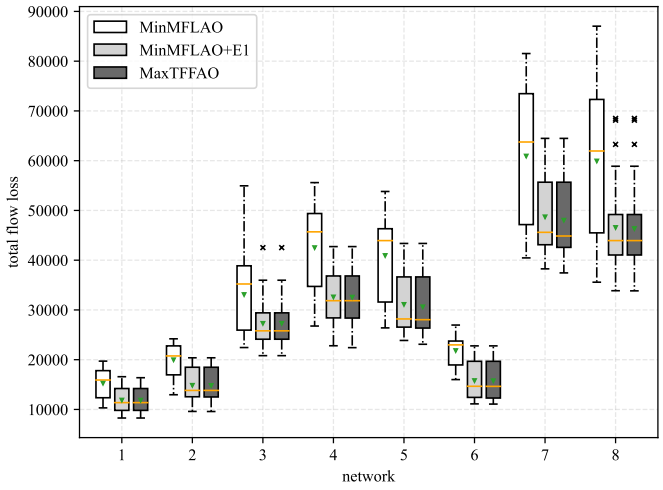


Figure 7: Comparison results on the total flow loss

FLFAO+E2" schedules are obtained by hierarchical optimization computing the MinMFLFAO in phase one and the extended problem E2 in phase two. Each box in the figure represents the performance of 30 instances for a network in terms of the duration of maximum flow loss. Particularly, for the MaxTFFAO schedules, the duration includes all time periods achieving or exceeding the maximum flow loss of the MinMFLFAO schedules.

As we can see, the MinMFLFAO schedules have many fewer time periods achieving the maximum flow loss, compared to the MaxTFFAO schedules. The duration of maximum flow loss in the "MinMFLFAO+E2" schedules is further reduced. Our hierarchical optimization approach is thus effective in scheduling preventive arc maintenance activities on networks to reduce the maximum flow loss and the duration of maximum flow loss.

7. Conclusions

In this work, we studied the network maintenance scheduling problem with flexible arc outages, aiming at finding a schedule with minimized maximum flow loss. We introduced a mixed integer programming formulation and a Benders reformulation for the problem. No-good cuts and goal cuts were proposed to reduce feasible regions for the master problem so that a branch-and-cut routine can be sped up. A Benders decomposition algorithm within a branch-and-cut framework was developed and improved with strong initial optimality cuts and an efficient separation procedure. We introduced hierarchical optimization for further deciding the best schedule among multiple optimal schedules achieving the minimum maximum flow loss. In phase two of hierarchical optimization, two extended problems which respectively minimize

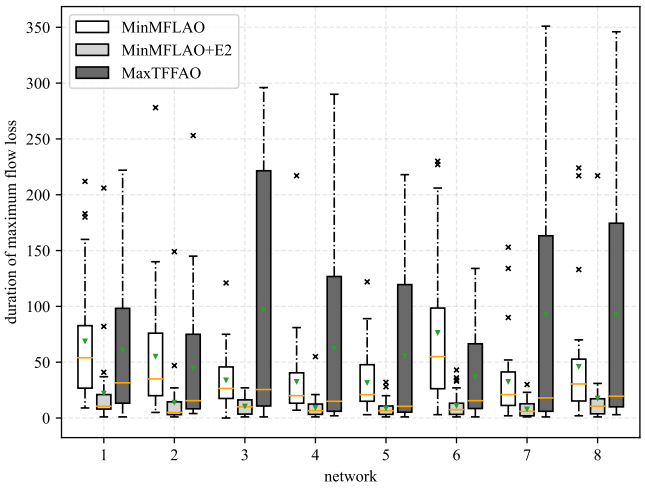


Figure 8: Comparison results on the duration of maximum flow loss

the total flow loss and the duration of maximum flow loss were studied. Our algorithm can also be adapted to solve well these two extensions. Computational experiments were carried out on a set of benchmark instances and a set of newly generated simulated instances, the results verified the effectiveness of the proposed algorithm. It solved all benchmark instances to optimality and was much faster than applying a mathematical programming solver to the mixed integer programming formulation and an existing Benders decomposition algorithm in the literature. On the simulated instances, our algorithm also outperforms the other two methods in terms of both solution quality and computational efficiency. We evaluated the schedules of different problem variants in terms of flow loss, total flow loss, duration of maximum flow loss. On the whole, the proposed formulations and algorithm are effective in computing a satisfactory maintenance schedule where the worst flow loss is reduced. In future work, the problem can be extended by also considering the maintenance tasks on the nodes of the network. Our formulations and algorithm serve as a basis for such a study.

Acknowledgements

This research was supported by National Natural Science Foundation of China [Grants 72171111, 72394363, 72394360, and 71732003] and Postgraduate Research & Practice Innovation Program of Jiangsu Province [KYCX22_0062].

References

Bianchessi, N., Ángel Corberán, Plana, I., Reula, M., & Sanchis, J. M. (2022). The min-max close-enough arc routing problem. *European Journal of Operational Research*, *300*, 837–851. doi:https://doi.org/10.1016/j.ejor.2021.10.047.

Boland, N., Kalinowski, T., & Kaur, S. (2015). Scheduling network maintenance jobs with release dates and deadlines to maximize total flow over time: Bounds and solution strategies. *Computers & Operations Research*, *64*, 113–129. doi:10.1016/j.cor.2015.05.011.

Boland, N., Kalinowski, T., & Kaur, S. (2016). Scheduling arc shut downs in a network to maximize flow over time with a bounded number of jobs per time period. *Journal of Combinatorial Optimization*, *32*, 885–905. doi:10.1007/s10878-015-9910-x.

Boland, N., Kalinowski, T., Waterer, H., & Zheng, L. (2014). Scheduling arc maintenance jobs in a network to maximize total flow over time. *Discrete Applied Mathematics*, *163*, 34–52. doi:10.1016/j.dam.2012.05.027.

Canto, S. P. (2008). Application of benders' decomposition to power plant preventive maintenance scheduling. *European Journal of Operational Research*, *184*, 759–777. doi:10.1016/j.ejor.2006.11.018.

Codato, G., & Fischetti, M. (2006). Combinatorial Benders' Cuts for Mixed-Integer Linear Programming. *Operations Research*, *54*, 756–766. doi:10.1287/opre.1060.0286.

Costa, A. M. (2005). A survey on benders decomposition applied to fixed-charge network design problems. *Computers & Operations Research*, *32*, 1429–1450. doi:https://doi.org/10.1016/j.cor.2003.11.012.

Crognier, G., Tournebise, P., Ruiz, M., & Panciatici, P. (2021). Grid operation-based outage maintenance planning. *Electric Power Systems Research*, *190*, 106682. doi:10.1016/j.epsr.2020.106682.

de Jonge, B., & Scarf, P. A. (2020). A review on maintenance optimization. *European Journal of Operational Research*, *285*, 805–824. doi:https://doi.org/10.1016/j.ejor.2019.09.047.

Ford, L., & Fulkerson, D. R. (1956). Maximal flow through a network. *Canadian Journal of Mathematics*, *8*.

Froger, A., Gendreau, M., Mendoza, J. E., Éric Pinson, & Rousseau, L.-M. (2016). Maintenance scheduling in the electricity industry: A literature review. *European Journal of Operational Research*, *251*, 695–706. doi:https://doi.org/10.1016/j.ejor.2015.08.045.

Goldberg, A. V., & Tarjan, R. E. (1986). A new approach to the maximum flow problem. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing* (p. 136–146). New York, NY, USA: Association for Computing Machinery. doi:10.1145/12130.12144.

Goldfarb, D., & Grigoriadis, M. D. (1988). A computational comparison of the dinic and network simplex methods for maximum flow. *Annals of Operations Research*, *13*, 81–123. doi:10.1007/BF02288321.

Gurobi Optimization Inc (2024). Gurobi optimizer reference manual. [Online]. URL: https://www.gurobi.com.

Hartmanis, J. (1982). Computers and Intractability: A Guide to the Theory of NP-Completeness (Michael R. Garey and David S. Johnson). *SIAM Review*, *24*, 90–91. doi:10.1137/1024022.

Hooker, J. (2000). *Logic-Based Methods for Optimization: Combining Optimization and Constraint Satisfaction*. Hoboken, NJ, USA: John Wiley & Sons, Inc. doi:10.1002/9781118033036.

Jain, V., & Grossmann, I. E. (2001). Algorithms for Hybrid MILP/CP Models for a Class of Optimization Problems. *INFORMS Journal on Computing*, . doi:10.1287/ijoc.13.4.258.9733.

Kalinowski, T., Matthews, J., & Waterer, H. (2020). Scheduling of maintenance windows in a mining supply chain rail network. *Computers & Operations Research*, *115*, 104670. doi:10.1016/j.cor.2019.03.016.

Lin, G.-H., Yao, E.-Y., & He, Y. (1998). Parallel machine scheduling to maximize the minimum load with nonsimultaneous machine available times. *Operations Research Letters*, *22*, 75–81. doi:10.1016/S0167-6377(97)00053-9.

Magnanti, T. L., & Wong, R. T. (1981). Accelerating benders decomposition: Algorithmic enhancement and model selection criteria. *Operations Research*, *29*, 464–484. doi:10.1287/opre.29.3.464.

Michael, E., Wood, T. A., Manzie, C., & Shames, I. (2022). Sensitivity analysis for bottleneck assignment problems. *European Journal of Operational Research*, *303*, 159–167. doi:10.1016/j.ejor.2022.02.037.

Mäkinen, V., Staneva, V., Tomescu, A., Valenzuela, D., & Wilzbach, S. (2017). Interval scheduling maximizing minimum coverage. *Discrete Applied Mathematics*, *225*, 130–135. doi:10.1016/j.dam.2016.08.021.

Pearce, R. H., & Forbes, M. (2019). Disaggregated benders decomposition for solving a network maintenance scheduling problem. *Journal of the Operational Research Society*, *70*, 941–953. doi:10.1080/01605682.2018.1471374.

Rey, D., Bar-Gera, H., Dixit, V. V., & Waller, S. T. (2019). A branch-and-price algorithm for the bilevel network maintenance scheduling problem. *Transportation Science*, *53*, 1455–1478. doi:10.1287/trsc.2019.0896.

Rodríguez, J. A., Anjos, M. F., Côté, P., & Desaulniers, G. (2021). Accelerating benders decomposition for short-term hydropower maintenance scheduling. *European Journal of Operational Research*, *289*, 240–253. doi:10.1016/j.ejor.2020.06.041.

Tawarmalani, M., & Li, Y. (2011). Multi-period maintenance scheduling of tree networks with minimum flow disruption. *Naval Research Logistics (NRL)*, *58*, 507–530. doi:10.1002/nav.20455.

Thorsteinsson, E. S. (2001). Branch-and-Check: A Hybrid Framework Integrating Mixed Integer Programming and Constraint Logic Programming. In T. Walsh (Ed.), *Principles and Practice of Constraint Programming — CP 2001* (pp. 16–30). Berlin, Heidelberg: Springer Berlin Heidelberg volume 2239. doi:10.1007/3-540-45578-7_2.