Annals of Operations Research (2024) 338:269–302
https://doi.org/10.1007/s10479-023-05561-6

ORIGINAL RESEARCH

Bayer digestion maintenance optimisation with lazy constraints and Benders decomposition

Sandy Spiers[1,2] · Hoa T. Bui[1,2] · Ryan Loxton[1,2] · Moussa Reda Mansour[3] · Kylie Hollins[3] · Richard Francis[3] · Christopher Martindale[3] · Yogesh Pimpale[3]

Received: 5 September 2022 / Accepted: 8 August 2023 / Published online: 15 September 2023

Abstract
This paper describes a maintenance scheduling model for digester banks. Digester banks are network-connected assets that lie on the critical path of the Bayer process, a chemical refinement process that converts bauxite ore into alumina. The banks require different maintenance activities at different due times. Furthermore, the maintenance schedule is subject to production-related constraints and resource limitations. Given the complexity of scheduling maintenance for large fleets of digester banks, a continuous-time, mixed-integer linear program is formulated to find the cost-minimising maintenance schedule that satisfies all required constraints. A solution approach that employs lazy constraints and Benders decomposition is proposed to solve the model. Unlike generic implementations of Benders decomposition, we show that the subproblems can be solved explicitly using a specialist algorithm. We solve the scheduling model for realistic scenarios involving two Bayer refineries based in Western Australia.

Keywords Maintenance scheduling · Bayer digestion · Network assets · Benders decomposition · Lazy constraints

✉ Sandy Spiers
sandy.spiers@postgrad.curtin.edu.au

Hoa T. Bui
hoa.bui@curtin.edu.au

Ryan Loxton
r.loxton@curtin.edu.au

Moussa Reda Mansour
Moussa.Mansour@alcoa.com

1 ARC Centre for Transforming Maintenance through Data Science, Bentley, Australia

2 Curtin Centre for Optimisation and Decision Science, Curtin University, Perth, Australia

3 Alcoa of Australia Limited, Perth, Australia

1 Introduction

Refining operations are, by their nature, asset-intensive endeavours. Under harsh environmental and operational conditions, all assets and equipment inevitably face some form of degradation. Without maintenance interventions, degrading asset health can lead to safety concerns, equipment failure, and loss of production. Maintenance operations were once considered retroactive tasks conducted after a failure, but now proactive maintenance strategies have become mainstream within the resources industry and make up a substantial proportion of the operating costs of large-scale refineries (de Jonge & Scarf, 2020). Determining the optimal maintenance strategy can reduce running costs and enable more sustainable production, giving organisations a vital competitive advantage. This paper looks at building a maintenance scheduling model for digester banks, a critical asset used in the Bayer process.

The Bayer process is a chemical refinement process that converts bauxite ore into alumina. In the digestion phase of the Bayer process, bauxite slurry is mixed with hot caustic liquor in large banks of pressure vessels that act like pressure cookers (Li et al., 2015). Once processed, the slurry leaves the bank as supersaturated alumina in solution, also known as green liquor. This liquor is then passed onto the production units, where the Bayer process is continued.

In this paper, we outline the practical scheduling requirements for digester bank maintenance and formulate an appropriate scheduling model. We use a continuous time framework to ensure timing accuracy and robustness to increasing time horizons. To assist in solving the model at large dimensions, a Benders decomposition algorithm is introduced where the subproblems are solved using a specialist algorithm. Additionally, lazy constraints are used to better handle a proportion of the constraint set. Finally, model performance is evaluated on two real-life case studies involving Bayer refineries in Western Australia as well as several test instances.

1.1 Digester bank maintenance activities

Routine preventative maintenance activities are crucial to maintaining the health of the digester banks. The maintenance activities generally fall into two categories: bank cleanings and bank services. Due to a chemical reaction in the Bayer process, scale builds up on the digester equipment. Scale build-up is unavoidable and is often thick and very hard (Cheng et al., 2021). To manage scale build-up, digester banks require frequent cleanings. A cleaning activity involves taking a bank offline and mechanically removing the scale until it is all removed. Bank cleanings are relatively small maintenance activities that must be planned frequently to sustain bank health and avoid unplanned failures. On the other hand, a bank service is a major activity that may only occur following a bank cleaning. During a bank service, worn or broken components may be replaced or repaired. In practice, it is common for digester banks to be cleaned several times in between services.

Scale only builds up when a bank is operational, meaning maintenance due times are calculated based on operational time, not elapsed time. As services are major activities that occur less frequently than cleanings, it is common for a bank to be cleaned several times between services, and therefore have several distinct operational periods. Changing the schedule of cleanings affects a bank's operational time and hence changes when its next service is due. This presents a significant challenge for schedulers, as minor changes in the schedule of cleaning activities can lead to significant changes in the operational time of a bank, and hence when its next service should occur.

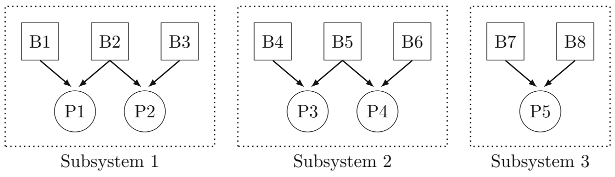

Fig. 1 Example digester setup with eight digester banks feeding five production units, split into three subsystems

A service can only occur immediately after a cleaning, meaning aligning these activities so they are due at similar times is beneficial. The best alignment of maintenance activities is a common challenge found in the resources sector, where assets may be subject to different levels of maintenance activities that are based on different cycle times (Seif et al., 2020). A trade-off must be made when the due times of two activities that must be scheduled together are misaligned. This compromise is often encountered when scheduling digester bank maintenance, as cleaning and service activities are rarely aligned, but must be completed together.

Digester banks sit on the refineries' critical path of production, meaning unplanned failures or downtimes can significantly impact site-wide production. To mitigate the risk of production loss due to digester bank failure, redundant digester banks are introduced. In a redundancy-based system, the number of available assets in a system exceeds the minimum required (Olde Keizer et al., 2016; Siopa et al., 2015). This allows production to be kept at full capacity, even when a bank is offline and receiving maintenance. Moreover, in the event of a failure of an operational bank, a standby bank can be brought online to rectify any potential production loss quickly. To demonstrate this, consider the example shown in Fig. 1, where the entire digestion system is split into several independent subsystems. In each subsystem, the number of available banks is more than production units, and hence not every bank is required to be operational at all times. From a scheduling perspective, managing the redundant digester banks is crucial. The redundant units allow the production load to be shared across all banks. This is a cost-effective strategy for refineries, as bank utilisation can be maximised without halting production. However, finding the optimal operational balance between banks and usage of the redundant bank is a challenging task.

Digester bank maintenance should align with the maintenance activities of the downstream production units, the most important of which is known as a *valve change*. Valve changes are relatively small activities that are essential to the operations of the entire refinery and are planned well in advance. After a valve change has been completed, the associated production unit must connect to a freshly cleaned digester bank. The difficulty with valve changes is that they are not always consistent, and are planned at fixed times. It is therefore very challenging to find a stable, repeatable maintenance pattern, as the schedule may be affected by an upcoming valve change. Subsequently, we need the maintenance schedule for digester banks to align with these significant activities.

Generally speaking, the revenue from production far outweighs the cost of preventative maintenance, meaning schedulers may be willing to over maintain to avoid the possibility of an unexpected failure. The tendency to over maintain is often exacerbated when the maintenance schedule is governed by complex or difficult-to-satisfy restrictions. This paper aims to build an optimisation model to plan the maintenance activities for a fleet of digester banks. Given the importance of digestion in the Bayer process, improvements in its maintenance strategy can lead to cost savings and improved sustainability of production.

1.2 Scheduling optimisation

The application of optimisation techniques to maintenance scheduling problems is a well-studied subject in the literature. For a recent review on maintenance optimisation including maintenance scheduling, the reader is directed to de Jonge and Scarf (2020). The vast majority of maintenance scheduling models can be separated into discrete or continuous-time models. In a discrete-time model, the desired planning or time horizon is broken up into many time windows or intervals. This allows for constraints to be formulated easily, as the schedule takes on a grid-like structure, thereby giving control to each discrete time point (Floudas & Lin, 2004). Discrete-time modelling has been successfully applied to many optimisation problems, such as job shop scheduling (Manne, 1960), the resource-constrained scheduling problem (Kopanos et al., 2014) and more complex parallel machine scheduling (Heydar et al., 2021). A significant challenge in discrete-time modelling is managing the dimension growth of time-indexed variables when the time horizon increases, or when greater time precision is required (Hooker, 2007). One method to overcome this difficulty is to move to a continuous-time model with discrete events.

Continuous-time models have often been proposed to overcome the challenges of their discrete-time counterparts (Kopanos et al., 2014). In continuous-time models, the decision variables are the continuous start times for the set of discrete events (Maravelias & Grossmann, 2003). This makes continuous time models more robust to increasing time horizons and can provide greater precision on event start times. Continuous-time modelling has been successfully applied to general problems such as the resource-constrained scheduling problem (Kopanos et al., 2014) and the multi-product batch process scheduling problem (Floudas & Lin, 2004; Maravelias & Grossmann, 2003), as well as to more complicated assign and schedule problems (Hooker, 2007). The continuous-time framework has recently been extended to event-based models. In event-based modelling, the time horizon is broken up into a set of events. Each event is then given a continuous variable denoting its start time, and binary variables are used to match activities to events. Koné et al. (2011) formulate two event-based models for the resource-constrained project scheduling problem and show that these models outperform others on a specific set of test instances. While continuous-time and event-based models are more stable when increasing the desired planning horizon, their scale increases with the number of discrete events that are to be scheduled. Furthermore, to attain the same level of control on the timing of events as discrete-time models, continuous-time models may require the inclusion of big-M constraints, which create poor LP relaxations (Maravelias & Grossmann, 2003). Overcoming these challenges can be achieved through the use of an appropriate solution approach.

A suitable solution approach can allow the model to be solved faster and at larger scales. The selection of solution approach often depends on the specific model structure, and the attributes that can be exploited. Lazy constraints are one such approach that can help solve problems that contain large constraint sets (Pearce & Forbes, 2018). The technique selects a subset of constraints called the lazy constraints, which are removed from the original problem to form a reduced problem and a set of lazy constraints. The reduced problem is then solved using a generic procedure, and whenever a solution is found, it is checked to see which lazy constraints are violated. Violated lazy constraints are then returned to the original problem. When the constraint set of the problem is large, but only a small portion are active in the optimal solution, the use of lazy constraints can substantially reduce the problem size and complexity. If only a tiny portion of lazy constraints must be reintroduced to the problem, then the solver benefits from a far simpler problem. This technique has successfully been applied to the classical traveling salesman problem (Miller et al., 1960), the resource-constrained scheduling problem (Lerch & Trautmann, 2019) and a network maintenance scheduling problem (Pearce & Forbes, 2018). For an in-depth analysis of the benefits of lazy constraints, the reader is directed to the PhD thesis Pearce (2019).

Another solution approach commonly used to assist in solving large scale scheduling problems is Benders decomposition. Benders decomposition is a partitioning technique that can break up a complex mixed-integer linear program (MILP) into easier to solve problems (Benders, 1962). Using Benders decomposition, the original problem is broken up into a mixed-integer master problem and several potentially independent continuous subproblems. The technique has been successfully applied to a wide range of optimisation problems. For an in-depth review of Benders decomposition, the reader is directed to Rahmaniani et al. (2017). When the subproblems are independent and easy to solve, Benders decomposition can lead to significant computational advantages. For example, in Fischetti et al. (2016), the Benders subproblem of the uncapacitated facility location problem were shown to reduce to the continuous knapsack problem, which permits a closed-form solution. Similarly, in Pearce and Forbes (2018), the subproblem of a network maintenance scheduling problem was shown to be equivalent to the minimum cut problem, for which many solution algorithms already exist. In some cases, the subproblems may exhibit a unique structure that can be solved using a specialist algorithm, thereby circumventing the need for an LP solver. Contreras et al. (2011) show that the subproblems of the uncapacitated hub location problem were semi-assignment problems that could be solved efficiently with a specialist algorithm. Similarly, in Naoum-Sawaya and Buchheim (2016), the subproblems of a critical node selection problem were solved using a specialist algorithm derived from the Floyd Warshall algorithm. Specialist algorithms such as these can provide significant computational improvements when solving the subproblems.

2 Problem formulation

The digester scheduling model should plan cleaning and service activities for a fleet of digester banks such that the maintenance cost is minimised and all maintenance, operational and production-related constraints are adhered to. We now describe the key constraints in detail, and outline the assumptions made for the scheduling model.

For this model, we assume that the digestion system consists of multiple independent subsystems, as in Fig. 1. Within each subsystem, we assume that there is always one redundant bank, i.e., the number of banks is one more than the number of production units. This means that exactly one bank is to be offline in each subsystem at all times. Additionally, we assume that a bank can only enter a standby period after completing a maintenance activity. Therefore, if a bank is to be taken offline for maintenance, the activity should be commenced immediately.

The due times for cleaning and service activities are based on operational time, not elapsed time. To ensure the model conducts maintenance on time, it must keep track of each bank's operational time and periods. In the case of cleaning activities, taking a bank offline requires that it immediately receives a cleaning and therefore, there is only ever one operational period between consecutive cleaning activities. However, this is not the case for service activities. A service is a significant activity that can only occur immediately after a cleaning activity, and has a longer operational due time than a cleaning activity. Therefore there will be several operational periods between consecutive service activities. The number of cleanings, and therefore operational periods, between consecutive service activities is unknown and may vary greatly depending on the specific bank and the required due times.

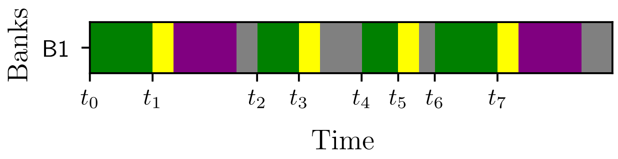

Fig. 2 Example schedule for a single digester bank. Green represents operational periods, whereas yellow and purple represent cleaning and services activities respectively. Grey represents periods where the bank is on standby

To demonstrate the complexity of operational due times, consider the example maintenance schedule for a single bank shown in Fig. 2. The green represents periods where the bank is operational, yellow represents cleaning activities, purple represents services, and grey represents periods where the bank is on standby. The due time of the cleaning at t_5 is not calculated as the elapsed time since the previous cleaning at t_3, but instead the period where the bank is operational, which is given as t_4 to t_5. Similarly, the service activity planned at t_7 is calculated based on the previous three operational periods since the last service at t_1, which are t_2 to t_3, t_4 to t_5 and t_6 to t_7. To ensure these activities are conducted on time, the model should keep track of each bank's operational time at all points in time.

The maintenance activities should be scheduled such that production levels are always satisfied, and resourcing restrictions are adhered to. To ensure required production levels are always met, there can only ever be one bank offline in each subsystem. This means that whenever a bank commences a maintenance activity, the offline bank in that subsystem must come back online to ensure that production levels are satisfied. However, a bank may not come back online until all its planned maintenance activities are completed.

We assume the labour force restrictions for both cleanings and services take two forms; overlap penalties and maximum available resources. A cost penalty is incurred whenever two cleanings (or services) are planned simultaneously. This is known as an overlap penalty. The penalty is incurred only for the period where the two activities overlap. Additionally, we assume that three or more simultaneous cleaning or service activities are not permitted, as this exceeds the maximum available resources. Note that it is possible to conduct, at most, two cleanings and two services simultaneously. While the model can easily be extended to consider a larger number of maximum available resources, this is considered beyond the scope necessary for practical implementations.

2.1 Model setup

The digester scheduling model is a continuous-time, mixed-integer linear program. For the schedule to be practical, it must plan a considerable amount into the future and decide on the start times with a high level of accuracy. For this reason, a discrete-time model is inappropriate, as the number of discrete time points required would be vast. Rather than being burdened by the enormous scale of a discrete-time model, a continuous-time model can span the same duration and provide a greater degree of accuracy with far fewer decision variables.

Given a set of available maintenance activities, the model should decide on the start time of each activity, which bank is to receive maintenance and whether the activity should be a cleaning or a service. For the entire digestion system, let $\mathcal{B}$ and $\mathcal{P}$ be the set of banks and production units respectively. We let $\mathcal{S}$ be the set of subsystems and hence, for every $s \in \mathcal{S}$, $\mathcal{B}_s \subset \mathcal{B}$ gives the subset of banks and $\mathcal{P}_s \subset \mathcal{P}$ gives the subset of production units within that

Table 1 Sets, parameters and decision variables

Sets	
$\mathcal{I}$	Set of available maintenance activities
$\mathcal{S}$	Set of subsystems
$\mathcal{B}$	Set of digester banks
$\mathcal{P}$	Set of production units
$\mathcal{D}$	Set of pairs of banks that lead to a double bank change
Parameters	
τ	Time horizon
α	Time to complete a cleaning
β	Time to complete a service
Θ	Operational due time of a cleaning
Φ	Operational due time of a service
A	Cost of completing a cleaning
B	Cost of completing a service
L	Penalty per day when planning simultaneous cleaning activities
M	Penalty per day when planning simultaneous service activities
Δ	Largest time between consecutive maintenance start times
Continuous decision variables	
t_i	Start time of maintenance activity $i \in \mathcal{I}$
$\theta_{i,b}$	Cleaning operational time for bank $b \in \mathcal{B}$ at the start of activity $i \in \mathcal{I}$
$\phi_{i,b}$	Service operational time for bank $b \in \mathcal{B}$ at the start of activity $i \in \mathcal{I}$
λ_i	Amount of cleaning overlap for activity $i \in \mathcal{I}$
μ_i	Amount of service overlap for activity $i \in \mathcal{I}$
Binary decision variables	
u_i	1 if maintenance activity $i \in \mathcal{I}$ is required
$x_{i,b}$	1 if maintenance activity $i \in \mathcal{I}$ is a cleaning on bank $b \in \mathcal{B}$
$y_{i,b}$	1 if maintenance activity $i \in \mathcal{I}$ is a service on bank $b \in \mathcal{B}$
$z_{i,b}$	1 if bank $b \in \mathcal{B}$ is remaining offline at the start of activity $i \in \mathcal{I}$

subsystem. Note that as each subsystem $s \in \mathcal{S}$ has exactly one extra bank, the cardinality of $\mathcal{B}_s$ is one more than $\mathcal{P}_s$. The model should determine the maintenance schedule up to a given time horizon, denoted by τ. To do so, let $\mathcal{I} = \{0, 1, \ldots, n\}$ be the set of available maintenance activities. The model need not use all available maintenance activities, in fact, we intentionally provide an overestimate of the number of maintenance activities needed, allowing the model to decide exactly how many are required. The decision variables and remaining parameters are defined as required when formulating the constraint set, which is done in the remainder of this section. All sets, parameters and decision variables are summarised in Table 1.

2.2 Scheduling maintenance activities

Let t_i be a continuous variable that gives the start time of maintenance activity $i \in \mathcal{I}$. As the set $\mathcal{I}$ is an overestimate of the number of activities required, let binary decision variable u_i equal 1 if maintenance activity $i \in \mathcal{I}$ is required, and 0 otherwise. Let binary decision variable $x_{i,b}$ equal 1 if activity $i \in \mathcal{I}$ is a cleaning on bank $b \in \mathcal{B}$. As a service can only ever occur immediately after a cleaning activity, we group these into one variable. Hence, let binary decision variable $y_{i,b}$ equal 1 if activity $i \in \mathcal{I}$ is a service on bank $b \in \mathcal{B}$. Within this activity, the cleaning is completed first and immediately after it the service is commenced. To track a bank's operational periods, let binary variable $z_{i,b}$ equal 1 if bank b has not yet been restarted by activity $i \in \mathcal{I}$, following a recently completed maintenance activity.

The timing of the maintenance activities should be such that the schedule starts at zero, activities are completed in order, and the schedule lasts the time horizon, i.e.,

$$t_0 = 0, \qquad t_i \geq t_{i-1}, \qquad t_n \geq \tau, \qquad i = 1, \ldots, n. \tag{1}$$

The use of $t_0 = 0$ ensures the schedule commences immediately, and thus by having $t_n \geq \tau$, the schedule is guaranteed to last for the desired time horizon. The model should decide exactly how many maintenance activities it requires, and use these up in order until no more are required. This is formulated as

$$u_i \leq u_{i-1}, \quad i = 1, \ldots, n. \tag{2}$$

Hence, once $u_i = 0$, none of the remaining activities will be used. If $u_i = 1$, then a maintenance activity must be started, therefore,

$$\sum_{b \in \mathcal{B}} \left(x_{i,b} + y_{i,b}\right) = u_i, \quad \forall i \in \mathcal{I}. \tag{3}$$

At any point in time, the number of banks offline, either receiving maintenance or on standby, is equal to number of subsystems (as there is one extra bank in each subsystem). Therefore, whenever an activity is started, there will be several banks remaining offline, either continuing a maintenance activity, or on standby after a recently completed activity. As these banks are offline, this period should not count towards their operational due times. To keep track of the periods that a bank remains offline after beginning maintenance, but before restarting, let binary variable $z_{i,b}$ equal 1 if bank b is yet to be restarted following a recent maintenance, at the start of activity $i \in \mathcal{I}$. Then $z_{i,b}$ may be updated with the following constraints,

$$z_{i,b} + x_{i,b} + y_{i,b} \leq 1, \qquad \forall b \in \mathcal{B}, i \in \mathcal{I}, \tag{4}$$

$$z_{i,b} \leq x_{i-1,b} + y_{i-1,b} + z_{i-1,b}, \qquad \forall b \in \mathcal{B}, i = 1, \ldots, n, \tag{5}$$

$$z_{i,b} \leq 1 - \sum_{b' \in \mathcal{B}_s} \left(x_{i,b'} + y_{i,b'}\right), \qquad \forall s \in \mathcal{S}, b \in \mathcal{B}, i \in \mathcal{I}, \tag{6}$$

$$x_{i,b} + y_{i,b} \leq 1 - x_{i-1,b} - y_{i-1,b}, \qquad \forall b \in \mathcal{B}, i = 1, \ldots, n, \tag{7}$$

$$\sum_{b \in \mathcal{B}} z_{i,b} = |\mathcal{S}| - u_i, \qquad \forall b \in \mathcal{B}, i \in \mathcal{I}. \tag{8}$$

Constraint (4) ensures that each bank is in at most one state in each period and (5) ensures that $z_{i,b} = 0$ if in the previous period bank b was not cleaned, serviced or remaining offline. Constraint (6) ensures that a bank cannot remain on standby if a new maintenance activity is planned in the same subsystem (and therefore forcing the bank to come online). Constraint (7) ensures that the same bank does not have maintenance activities planned in consecutive activities. Lastly, constraint (8) ensures that the correct number of banks are chosen to have $z_{i,b} = 1$.

Note that for the purpose of our model, the schedule is assumed to start immediately, with the first activity planned at $t_0 = 0$. However, in practice, the scheduling of the first maintenance activity is often influenced by the current state of the bank setup. This includes situations where the schedule should begin with banks on standby, rather than immediately starting a maintenance activity. In such cases, constraints (3) and (8) can be modified to exclude the case where $i = 0$ and thus allow $x_{0,b} = y_{0,b} = 0$ for all $b \in \mathcal{B}$. Additionally, the values of $z_{0,b}$ can be constrained appropriately to match the banks that are currently offline. This allows the model to begin with the correct banks on standby, without affecting subsequent maintenance intervals.

Let α be the time to complete a cleaning activity, and let β be the time to complete a full service, including the cleaning. As a service is a major activity that has a cleaning as one of its subtasks, we assume that $\alpha < \beta$. Maintenance activities must be timed such that there is only ever one bank in each subsystem offline at any time, thus ensuring production levels are always maintained. Additionally, across all banks there may never be three or more simultaneous cleanings or services. Finally, the timing of maintenance activities should account for overlapping activities, such that the overlap can be included in the objective function as a penalty.

Consider first the timing of cleaning activities. As a service always begins with a cleaning activity, a cleaning is undergone at the start of every used maintenance activity. Therefore, to avoid three simultaneous cleanings, whenever an activity i is planned for maintenance, we must have $t_{i+2} \geq t_i + \alpha$, as this ensures the cleaning in i has finished, before the cleaning in $i + 2$ starts. This can formulated as so,

$$t_i \geq t_{i-2} + \alpha u_{i-2}, \quad i = 2, \ldots, n. \tag{9}$$

To avoid overlapping cleanings within the same subsystem, consecutive activities may not overlap if they are planned for the same subsystem. This can be formulated as,

$$t_i \geq t_{i-1} + \alpha \left(\sum_{b \in \mathcal{B}_s} \left(x_{i-1,b} + y_{i-1,b} + x_{i,b} + y_{i,b} \right) - 1 \right), \quad \forall s \in \mathcal{S}, i = 1, \ldots, n. \tag{10}$$

Whenever consecutive maintenance activities are planned for the same subsystem, i.e., $\sum_{b \in \mathcal{B}_s} \left(x_{i-1,b} + y_{i-1,b} + x_{i,b} + y_{i,b} \right) = 2$, constraint (10) becomes $t_i \geq t_{i-1} + \alpha$. Therefore, the cleaning must be completed before the next one is started, thus maintaining production levels.

To introduce the overlap penalties for cleaning activities, let λ_i be a continuous variable that gives the amount of overlap between the cleaning activities in $i - 1$ and i. Then

$$0 \leq \lambda_i \leq \alpha, \quad \forall i \in \mathcal{I}. \tag{11}$$

The model may decide how much cleaning overlap to accept with the following constraint,

$$t_i \geq t_{i-1} + \alpha u_{i-1} - \lambda_i, \quad i = 1, \ldots, n. \tag{12}$$

Whenever the cleaning component of activities i and $i - 1$ overlap, λ_i will get the amount of overlap. This is then included as a penalty in the objective function.

Avoiding clashes in service activities is slightly more challenging as a service activity may span several maintenance activities, however the constraints can be formulated in an analogous way to (9), (10) and (12), and are give as

$$t_j \geq t_i + \beta \left(\sum_{b \in \mathcal{B}_s} \left(y_{i,b} + x_{j,b} + y_{j,b} \right) - 1 \right), \qquad \forall s \in \mathcal{S}, i, j \in \mathcal{I} : i < j, \tag{13}$$

$$t_j \geq t_i + \beta \left(\sum_{b \in \mathcal{B}_s} \left(y_{i,b} + y_{j,b} \right) - 1 \right) - \alpha - \mu_j, \qquad \forall s \in \mathcal{S}, i, j \in \mathcal{I} : i < j, \tag{14}$$

$$t_k \geq t_i + \beta \left(\sum_{b \in \mathcal{B}_s} \left(y_{i,b} + y_{j,b} + y_{k,b} \right) - 2 \right) - \alpha, \qquad \forall s \in \mathcal{S}, i, j, k \in \mathcal{I} : i < j < k. \tag{15}$$

Constraint (13) ensures that no activity is commenced in a subsystem until enough time has passed since the last conducted service in that subsystem, similar to constraint (10). Constraint (14) ensures that whenever two activities have services that overlap, μ_j is forced to get the amount of overlap, which can then be included in the objective function as a cost penalty. As a service includes the cleaning at the start of the activity, and goes into the service immediately afterwards, this should be offset by α such that the overlap only considers the service component. Finally, constraint (15) ensures that three or more services are never planned simultaneously. While the number of constraints introduced here is large, few are expected to be binding in the optimal solution, as services are major activities that are sparsely scheduled.

2.3 Maintenance due times

To ensure banks are maintained on time, the model should count the time each bank is operational for. Let $\theta_{i,b}$ and $\phi_{i,b}$ be bank b's operational time since its last cleaning and service respectively, at the start of activity $i \in \mathcal{I}$. Then, let Θ and Φ be the operational due times of a cleaning and service activity, respectively. As a service is a major activity, we assume that $\Theta < \Phi$. In practice, we expect Φ to be several times greater than Θ. Lastly, let $\tilde{\theta}_b$ and $\tilde{\phi}_b$ give the starting operational time for bank $b \in \mathcal{B}$ since its last cleaning or service respectively, and let Δ be a parameter such that $\Delta \geq t_i - t_{i-1}$ for $i = 1, \ldots, n$. Note that we can always choose $\Delta = \Theta$.

Observe that the model does not require the exact values of θ and ϕ at every maintenance activity, it only requires that maintenances are scheduled on time. Furthermore, the model has no benefit from having large values of θ and ϕ, as this leads to more required maintenance activities due to longer bank runtimes. For this reason, rather than forcing the model to calculate the exact values of θ and ϕ, we can instead provide exact lower bounds. For ϕ, this can be achieved with,

$$\phi_{0,b} = \tilde{\phi}_b, \qquad \forall b \in \mathcal{B}, \tag{16}$$

$$\begin{aligned} \phi_{i,b} \geq \phi_{i-1,b} + t_i - t_{i-1} - (\Phi + \Delta)\, y_{i-1,b} \\ - \Delta \left(x_{i-1,b} + z_{i-1,b} \right), \end{aligned} \qquad \forall b \in \mathcal{B}, i = 1, \ldots, n, \tag{17}$$

$$\phi_{i,b} \geq \phi_{i-1,b} - \Phi y_{i-1,b}, \qquad \forall b \in \mathcal{B}, i = 1, \ldots, n, \tag{18}$$

$$0 \leq \phi_{i,b} \leq \Phi, \qquad \forall b \in \mathcal{B}, i \in \mathcal{I}. \tag{19}$$

Any feasible solution satisfies (17–19), and therefore

$$\phi_{i,b} \geq \max \left\{ \phi_{i-1,b} + t_{i+1} - t_i - (\Phi + \Delta)\, y_{i-1,b} - \Delta \left(x_{i-1,b} + z_{i-1,b} \right), \phi_{i-1,b} - \Phi y_{i-1,b}, 0 \right\}$$

for $i = 1, \dots, n$. If, during the previous activity, the bank was operational, then $x_{i-1,b} + y_{i-1,b} + z_{i-1,b} = 0$ and hence

$$\phi_{i,b} \geq \max\left\{\phi_{i-1,b} + t_{i+1} - t_i, \phi_{i-1,b}, 0\right\} = \phi_{i-1,b} + t_{i+1} - t_i,$$

as required. Alternatively, if the bank was cleaned or on standby, then $x_{i-1,b} + z_{i-1,b} = 1$ and $y_{i-1,b} = 0$ and hence

$$\phi_{i,b} \geq \max\left\{\phi_{i-1,b} + t_{i+1} - t_i - \Delta, \phi_{i-1,b}, 0\right\} = \phi_{i-1,b},$$

as required. Finally, if the bank was serviced, then $x_{i-1,b} + z_{i-1,b} = 0$ and $y_{i-1,b} = 1$ and hence

$$\phi_{i,b} \geq \max\left\{\phi_{i-1,b} + t_{i+1} - t_i - \Phi - \Delta, \phi_{i-1,b} - \Phi, 0\right\} = 0,$$

as required. The same logic can then be applied to θ,

$$\begin{aligned}
&\theta_{0,b} = \tilde{\theta}_b, && \forall b \in \mathcal{B}, && (20)\\
&\theta_{i,b} \geq \theta_{i-1,b} + t_i - t_{i-1} - (\Theta + \Delta)\left(x_{i-1,b} + y_{i-1,b}\right) && && \\
&\quad - \Delta z_{i-1,b}, && \forall b \in \mathcal{B}, i = 1, \dots, n, && (21)\\
&0 \leq \theta_{i,b} \leq \Theta, && \forall b \in \mathcal{B}, i \in \mathcal{I}. && (22)
\end{aligned}$$

This provides an appropriate formulation to update the exact lower bounds on θ and ϕ, and thus ensure banks are maintained on time.

2.4 Double bank changes

Bank switching may only take place between banks that can connect to the same production unit. For example, consider subsystem 1 of Fig. 1. Bank 1 may not be switched with bank 3, as they do not share a common production unit. However bank 1 may be switched with bank 2. Switching banks 1 and 3 is known as a *double bank switch*, and must be avoided. Let $\mathcal{D}_s$ be the set of pairs of banks in subsystem $s \in \mathcal{S}$ that cannot connect to the same production unit. For instance, $(b_1, b_2) \in \mathcal{D}_s$ if $b_1, b_2 \in \mathcal{B}_s$ and they cannot connect to the same production unit. To avoid a double bank change, their maintenance activities should not take place in consecutive intervals. Hence,

$$x_{i,b_1} + y_{i,b_1} \leq 1 - x_{i-1,b_2} - y_{i-1,b_2} - z_{i-1,b_2}, \quad \forall s \in \mathcal{S}, (b_1, b_2) \in \mathcal{D}_s, i = 1, \dots, n. \tag{23}$$

2.5 Estimating number of required maintenance activities

An appropriate upper bound for the number of maintenance activities required to span a time horizon is given by total number of activities that could potentially be scheduled. This is given by the schedule that assumes constant cleaning activities with full overlap. Let $f : \mathbb{R} \to \mathbb{Z}$ be an integer-valued function that determines an upper bound on the number of activities required to span a given time horizon. Then,

$$f(t) := 2\left\lfloor \frac{t}{\alpha} \right\rfloor, \tag{24}$$

where $t \geq 0$ is the time frame the schedule should span.

2.6 Valve changes

On every valve change, the schedule requires that a freshly cleaned digester bank is connected to the associated production unit. To account for valve changes, we can break the schedule up into distinct time windows, separated by the planned valve changes. Within each window a set of maintenance activities are provided that can be used to schedule activities. As before, this set of activities is an over-estimate of the number required. The model should then use these maintenance activities to find a schedule that starts exactly on a valve change, and finishes at the next.

Let $V = (v_1, \dots, v_m)$ be a vector of m planned valve changes, such that $v_1 < v_2 < \cdots < v_m < \tau$ and let $P = (p_1, \dots, p_m)$ denote the associated production unit of each valve change. The time horizon can be broken up into $m + 1$ distinct time windows, denoted by the set $\mathcal{W} = \{1, \dots, m+1\}$. The first m time windows end at the next valve change, while the last window ends at τ. Let $n_1 = f(v_1)$, $n_w = f(v_w - v_{w-1})$ for $w = 2, \dots, m$, and $n_{m+1} = f(\tau - v_m)$ be the number of activities required to span each time window. Let $\mathcal{I} = \{0, \dots, n\}$ be defined as before, except now let $n = \sum_{w \in \mathcal{W}} n_w$. Lastly, let $\mathcal{J} = \{0, n_1, n_1 + n_2, \dots, n_1 + \cdots + n_m\}$ be the set that denotes the starting activity of each time window. Whereas in the previous formulation the model used up maintenance activities until no more were required over the entire time horizon, this should be amended to within each time window. To formulate this, constraint (2) should be updated to the following,

$$u_i = 1, \qquad \forall i \in \mathcal{J}, \tag{25}$$

$$u_i \leq u_{i-1}, \qquad \forall i \in \mathcal{I} \setminus \mathcal{J}. \tag{26}$$

Therefore $u_i = 1$ for the first activity of every time window, given by $i \in \mathcal{J}$. Within each time window, activities are used up until no more are required.

Now that it is known what activities are to be planned at valve changes, the following constraints ensure the schedule adheres to the valve change,

$$t_i = v_w, \qquad w = 1, \dots, m, i = \sum_{j=1}^{w} n_j, \tag{27}$$

$$\sum_{b \in \mathcal{B}_{p_w}} \left(x_{i,b} + y_{i,b} \right) \geq 1, \qquad w = 1, \dots, m, i = \sum_{j=1}^{w} n_j, \tag{28}$$

$$\sum_{b \in \mathcal{B}_{p_w}} \left(x_{i-1,b} + y_{i-1,b} + z_{i-1,b} \right) \geq 1, \qquad w = 1, \dots, m, i = \sum_{j=1}^{w} n_j, \tag{29}$$

where $\mathcal{B}_{p_w} \subset \mathcal{B}$ gives the subset of banks that can connect to production unit $p_w \in \mathcal{P}$. Constraint (27) ensures that the activities associated with valve changes are time appropriately. Constraint (28) ensures that the bank planned for maintenance in $i \in \mathcal{J}$ can connect to the appropriate production unit. Similarly, constraint (29) ensures that in the previous activity the bank offline can also connect to the appropriate production unit. In tandem, these constraints ensure a bank switch is occurring on the production unit, thus meeting the requirements of a valve change.

2.7 Objective function

The objective of this model is to determine the maintenance schedule that optimises cost. This cost is made up of planned maintenance activities as well as overtime cost from overlapping activities. Let A and B be the cost of a cleaning and service activity respectively and let L and M be the additional cost per day of operating two cleaning and service activities respectively. The full model, denoted by OP, can then be formulated as so,

$$
\begin{aligned}
[OP] \quad & \min \sum_{i\in\mathcal{I}} \left(\sum_{b\in\mathcal{B}} \left(A x_{i,b} + B y_{i,b} \right) + L\lambda_i + M\mu_i \right), \\
& \text{subject to } (1), (3) - (23), (25) - (29), \\
& \qquad x_{i,b}, y_{i,b}, z_{i,b} \in \{0,1\}, \quad \forall i \in \mathcal{I}, b \in \mathcal{B}, \\
& \qquad u_i \in \{0,1\}, \quad \forall i \in \mathcal{I}, \\
& \qquad \theta_{i,b}, \phi_{i,b} \geq 0, \quad \forall i \in \mathcal{I}, b \in \mathcal{B}, \\
& \qquad t_i, \lambda_i, \mu_i \geq 0, \quad \forall i \in \mathcal{I}.
\end{aligned}
$$

3 Solution algorithm

The original problem OP is solved using a combination of lazy constraints and Benders decomposition. Additionally, valid inequalities are introduced to further tighten the problem formulation.

Lazy constraints are a modelling technique that removes a subset of constraints from the original problem to form a reduced problem. The lazy constraints are then added back to the reduced problem only when the solver deems that the constraint is necessary. If only a small fraction of the lazy constraints are required to be added back in order to find the optimal solution, then the solver benefits from a far simpler problem, as a large number of unnecessary constraints have been removed.

Services are significant activities that are expected to be planned infrequently. In many practical examples, the total number of services planned may be as little as one-fifth of the total number of cleaning activities. However, the constraints required to ensure service clashes, in particular (15), represent a significant proportion of total constraints for the model. As only a few services are expected to be scheduled, the number of active constraints in this set is small. Therefore, by formulating (15) as lazy constraints, we are able to reduce the constraint set to only those that are necessary.

Benders decomposition is a partitioning technique that can break up the original problem into a mixed-integer master problem and several potentially independent continuous subproblems. Once a solution to the master problem is found, this solution is used to construct dual subproblems. By solving the dual subproblems, feasibility and optimality cuts may be generated and added to the master problem to either remove this solution, or improve the objective value.

By applying Benders decomposition to the variables θ and ϕ, we construct the *lifetime subproblems* using the maintenance due time constraints (16–22). These subproblems are used to identify periods in which a bank is overrun with respect to its cleaning and service lifetimes. As such, the lifetime subproblems are purely feasibility problems. The subproblems can be solved independently by separating them into each bank's cleaning and service lifetime subproblem. If a subproblem is infeasible, i.e., a bank is overrun, appropriate feasibility cuts are added to the master problem to remove this solution and ensure the bank is maintained on time. Finally, to tighten the master problem formulation, a set of valid inequalities based on practical assumptions of the problem is introduced.

3.1 Master problem

The master problem (denoted by MP) follows a natural interpretation. It determines a candidate maintenance schedule that minimises the total cost due to planned maintenance activities and overlap penalties. The schedule should avoid double bank changes and be aligned with the planned valve changes. Additionally, the problem should satisfy a set of valid inequalities, added to tighten the master problem formulation. Lazy constraints are added to MP whenever a solution that violates a constraint in (15) is found. Finally, feasibility cuts are added whenever a schedule does not maintain a bank on time. Therefore, the master problem can be formulated as so,

$$
\begin{aligned}
[MP] \quad & \min \sum_{i \in \mathcal{I}} \left(\sum_{b \in \mathcal{B}} \left(A x_{i,b} + B y_{i,b} \right) + L\lambda_i + M\mu_i \right), \\
& s.t.\ (1), (3) - (14), (23), (25) - (29), \\
& \qquad \text{Lazy Constraints}, \\
& \qquad \text{Feasibility Cuts}, \\
& \qquad \text{Valid Inequalities}, \\
& \qquad x_{i,b}, y_{i,b}, z_{i,b} \in \{0,1\}, \quad \forall i \in \mathcal{I}, b \in \mathcal{B}, \\
& \qquad u_i \in \{0,1\}, \quad \forall i \in \mathcal{I}, \\
& \qquad t_i, \lambda_i, \mu_i \geq 0, \quad \forall i \in \mathcal{I},
\end{aligned}
$$

where the feasibility cuts and valid inequalities are described in the following sections.

3.2 Lifetime subproblems

The lifetime subproblems determine whether a candidate solution of MP overruns any of the banks with respect to either cleaning or service activities. If a bank is overrun, a feasibility cut is generated and added to MP to remove this solution. The subproblems are broken up into cleaning lifetimes (associated with variable θ) and service lifetimes (associated with variable ϕ) and then separated further by each bank $b \in \mathcal{B}$. Therefore the number of lifetime subproblems is $2 \times |\mathcal{B}|$. In general, feasibility cuts require using an LP solver to determine an extreme ray. Here, we show that the extreme rays of the lifetime subproblems can be generated using the exact operational time of each bank, thereby circumventing the need for an LP solver.

We begin this section by formulating the dual of the cleaning and service lifetime subproblems. We then show how the candidate schedule from the master problem can be used to calculate the exact operational time of each bank. The exact operational time of each bank is then used as a candidate solution for its lifetime subproblems. Notably, we show how a subproblem is feasible if and only if the exact operational time is feasible. Furthermore, this candidate solution can then be used to generate extreme rays, and thus feasibility cuts.

Given a fixed schedule $(x_{i,b}, y_{i,b}, z_{i,b}, t_i)$ for all $b \in \mathcal{B}$, $i \in \mathcal{I}$, the cleaning lifetime subproblems attempt to find values for $\theta_b \in \mathbb{R}^{n+1}$ such that constraints (20–22) are satisfied for each bank $b \in \mathcal{B}$. The cleaning lifetime subproblem of bank $b \in \mathcal{B}$ (denoted by θ_b-SP) is formulated as follows,

$$
\begin{aligned}
[\theta_b\text{-}SP] \quad & \min \quad 0 && (30)\\
s.t. \quad & \theta_0 \geq \tilde{\theta}, && (31)\\
& \theta_i - \theta_{i-1} \geq t_i - t_{i-1} - (\Theta + \Delta)\left(x_{i-1,b} + y_{i-1,b}\right) - \Delta z_{i-1,b}, \quad i = 1, \ldots, n, && (32)\\
& \theta_i \leq \Theta, \quad i = 0, \ldots, n && (33)\\
& \theta_i \geq 0, \quad i = 0, \ldots, n. && (34)
\end{aligned}
$$

The dual problem of θ_b-SP can be written in terms of dual variables $\gamma \in \mathbb{R}^{n+1}$ and $\eta \in \mathbb{R}^{n+1}$ and is denoted by θ_b-DP,

$$
\begin{aligned}
[\theta_b\text{-}DP] \quad \max \quad & \tilde{\theta}\gamma_0 + \sum_{i=1}^{n}\left(t_i - t_{i-1} - (\Theta + \Delta)\left(x_{i-1,b} + y_{i-1,b}\right) - \Delta z_{i-1,b}\right)\gamma_i + \sum_{i=0}^{n}\Theta\eta_i\\
s.t. \quad & \gamma_i - \gamma_{i+1} + \eta_i \leq 0, \quad i = 0, \ldots, n-1,\\
& \gamma_n + \eta_n \leq 0,\\
& \gamma_i \geq 0, \quad i = 0, \ldots, n,\\
& \eta_i \leq 0, \quad i = 0, \ldots, n.
\end{aligned}
$$

Similarly, the service lifetime subproblems attempt to find values for $\phi_b \in \mathbb{R}^{n+1}$ such that constraints (16–19) are satisfied for each bank $b \in \mathcal{B}$. The service lifetime subproblem of bank $b \in \mathcal{B}$ (denoted by ϕ_b-SP) is formulated as follows,

$$
\begin{aligned}
[\phi_b\text{-}SP] \quad & \min \quad 0 && (35)\\
s.t. \quad & \phi_0 \geq \tilde{\phi}, && (36)\\
& \phi_i - \phi_{i-1} \geq t_i - t_{i-1} - (\Phi + \Delta) y_{i-1,b} - \Delta\left(x_{i-1,b} + z_{i-1,b}\right), \quad i = 1, \ldots, n, && (37)\\
& \phi_i - \phi_{i-1} \geq -\Phi y_{i-1,b}, \quad i = 1, \ldots, n, && (38)\\
& \phi_i \leq \Phi, \quad i = 0, \ldots, n && (39)\\
& \phi_i \geq 0, \quad i = 0, \ldots, n. && (40)
\end{aligned}
$$

The dual problem of ϕ_b-SP can be written in terms of dual variables $\pi \in \mathbb{R}^{n+1}$, $\sigma \in \mathbb{R}^n$ and $\rho \in \mathbb{R}^{n+1}$ and is denoted by ϕ_b-DP,

$$
\begin{aligned}
[\phi_b\text{-}DP] \quad \max \quad & \Gamma_b = \tilde{\phi}\pi_0 + \sum_{i=1}^{n}\left(t_i - t_{i-1} - (\Phi + \Delta) y_{i-1,b} - \Delta\left(x_{i-1,b} + z_{i-1,b}\right)\right)\pi_i\\
& \qquad - \Phi\sum_{i=1}^{n} y_{i-1,b}\sigma_i + \sum_{i=0}^{n}\Phi\rho_i\\
s.t. \quad & \pi_0 - \pi_1 - \sigma_1 + \rho_0 \leq 0,\\
& \pi_i - \pi_{i+1} + \sigma_i - \sigma_{i+1} + \rho_i \leq 0, \quad i = 1, \ldots, n-1,\\
& \pi_n + \sigma_n + \rho_n \leq 0,\\
& \pi_i \geq 0, \quad i = 0, \ldots, n,\\
& \sigma_i \geq 0, \quad i = 1, \ldots, n,
\end{aligned}
$$

$$\rho_i \le 0, \quad i = 0, \ldots, n.$$

If for bank $b \in \mathcal{B}$, either θ_b-SP or ϕ_b-SP are infeasible, then the schedule generated by MP does not service the bank on time, and therefore feasibility cuts should be added to MP to remove this solution. To generate a feasibility cut, the exact operational times of each bank can be calculated, and provide all information required about θ_b-SP and ϕ_b-SP.

Definition 3.1 Let $\theta_b^* \in \mathbb{R}^{n+1}$ be the exact operational time of bank $b \in \mathcal{B}$ since its' last cleaning, based on the schedule determined by MP. This is the operational time of the bank at every$i \in \mathcal{I}$, if the schedule was followed in practice. Hence, the operational time increases whenever the bank is in operation, and resets to 0 whenever it is cleaned. Then θ_b^* can be calculated recursively as follows,

$$\theta_{i,b}^* := \begin{cases} \tilde{\theta}_b, & i = 0, \\ \left(\theta_{i-1,b}^* + t_i - t_{i-1}\right)\left(1 - x_{i-1,b} - y_{i-1,b} - z_{i-1,b}\right), & i = 1, \ldots, n. \end{cases} \tag{41}$$

Similarly, let $\phi_b^* \in \mathbb{R}^{n+1}$ be the exact operational time of bank $b \in \mathcal{B}$ since its' last service, based on the schedule determined by MP. Then ϕ_b^* can be calculated recursively as follows,

$$\phi_{i,b}^* := \begin{cases} \tilde{\phi}_b, & i = 0 \\ \phi_{i-1,b}^*\left(1 - y_{i-1,b}\right) + \left(t_i - t_{i-1}\right)\left(1 - x_{i-1,b} - y_{i-1,b} - z_{i-1,b}\right), & i = 1, \ldots, n. \end{cases} \tag{42}$$

Lemma 3.2 *Let $\phi_b \in \mathbb{R}^{n+1}$ be such that it satisfies (36) - (38) and (40), and let $\phi_{i,b}^*$ be defined as in (42). Then $\phi_{i,b} \ge \phi_{i,b}^*$ for all $i \in \mathcal{I}$.*

Proof We prove $\phi_i \ge \phi_{i,b}^*$ for all $i \in \mathcal{I}$ by induction on dimension i. The base case $i = 0$ holds because $\phi_{0,b}$ satisfies (36) and therefore $\phi_{0,b} \ge \tilde{\phi}_b = \phi_{0,b}^*$. Suppose $\phi_{i,b} \ge \phi_{i,b}^*$ holds for $i \le n-1$. We now prove that $\phi_{i+1,b} \ge \phi_{i+1,b}^*$ holds by considering three cases. From constraint (4) of MP either $x_{i,b} = y_{i,b} = z_{i,b} = 0$, or $x_{i,b} + z_{i,b} = 1$ and $y_{i,b} = 0$, or finally $y_{i,b} = 1$ and $x_{i,b} + z_{i,b} = 0$. As $\phi_{i,b}$ satisfies (37), (38) and (40) we have that

$$\phi_{i+1,b} \ge \max\left\{0, \phi_{i,b} + t_{i+1} - t_i - (\Phi + \Delta)\, y_{i,b} - \Delta\left(x_{i,b} + z_{i,b}\right), \phi_{i,b} - \Phi y_{i,b}\right\}. \tag{43}$$

Suppose firstly that $x_{i,b} = y_{i,b} = z_{i,b} = 0$, then from (43),

$$\begin{aligned} \phi_{i+1,b} &\ge \max\left\{0, \phi_{i,b} + t_{i+1} - t_i, \phi_{i,b}\right\} \\ &= \phi_{i,b} + t_{i+1} - t_i \\ &\ge \phi_{i,b}^* + t_{i+1} - t_i \\ &= \phi_{i+1,b}^*. \end{aligned}$$

Alternatively, suppose $x_{i,b} + z_{i,b} = 1$ and $y_{i,b} = 0$, then from (43),

$$\begin{aligned} \phi_{i+1,b} &\ge \max\left\{0, \phi_{i,b} + t_{i+1} - t_i - \Delta, \phi_{i,b}\right\} \\ &= \phi_{i,b} \\ &\ge \phi_{i,b}^* \\ &= \phi_{i+1,b}^*. \end{aligned}$$

Finally, suppose $y_{i,b} = 1$ and $x_{i,b} + z_{i,b} = 0$, then from (43),

$$\begin{aligned} \phi_{i+1,b} &\ge \max\left\{0, \phi_{i,b} + t_{i+1} - t_i - (\Phi + \Delta), \phi_{i,b} - \Phi\right\} \\ &= 0 \end{aligned}$$

$$= \phi^*_{i+1,b}.$$

Therefore, by induction, $\phi_{i,b} \geq \phi^*_{i,b}$ for all $i \in \mathcal{I}$. □

Lemma 3.3 *Let $\theta_b \in \mathbb{R}^n$ be such that it satisfies (31), (32), and (34), and let $\theta^*_{i,b}$ be defined as in (41). Then $\theta_{i,b} \geq \theta^*_{i,b}$ for all $i \in \mathcal{I}$.*

Proof The proof follows analogously from the proof of Lemma 3.2. □

We now show how θ^*_b and ϕ^*_b can be used to precisely determine the feasibility of θ_b-SP and ϕ_b-SP. The proofs are shown to hold for ϕ^* first, as this is the more complicated subproblem and the proofs for θ^* follow analogously.

Proposition 3.4 *ϕ_b-SP is feasible if and only if $\phi^*_{i,b} \leq \Phi$ for all $i \in \mathcal{I}$.*

Proof We first prove the forward statement. If ϕ_b-SP is feasible, then there exists a feasible solution $\phi_{i,b}$ that satisfies (36–40). From Lemma 3.2 we therefore have $\phi_{i,b} \geq \phi^*_{i,b}$ for all $i \in \mathcal{I}$. As $\phi_{i,b}$ satisfies (39), we have that $\phi_{i,b} \leq \Phi$ for all $i \in \mathcal{I}$ and therefore $\phi^*_{i,b} \leq \Phi$ for all $i \in \mathcal{I}$.

We now prove the reverse statement, by showing that if $\phi^*_{i,b} \leq \Phi$ for all $i \in \mathcal{I}$, then $\phi^*_{i,b}$ is a feasible solution to ϕ_b-SP. Firstly, if $\phi^*_{i,b} \leq \Phi$ for all $i \in \mathcal{I}$, then ϕ^*_b satisfies (39). From (42), we have that $\phi^*_{0,b} \geq \tilde{\phi}_b$, thereby satisfying (36). We now prove that $\phi^*_{i,b}$ satisfies (40) by induction on dimension i. The base case $i = 0$ holds because $\phi^*_{0,b} \geq \tilde{\phi}_b \geq 0$. Suppose $\phi^*_{i,b} \geq 0$ holds for $i \leq n-1$. We now prove that $\phi^*_{i+1,b} \geq 0$ also holds. Recall from (42),

$$\phi^*_{i+1,b} = \phi^*_{i,b}\left(1 - y_{i,b}\right) + \left(t_{i+1} - t_i\right)\left(1 - x_{i,b} - y_{i,b} - z_{i,b}\right). \tag{44}$$

From constraint (1) we have $t_i - t_{i-1} \geq 0$. Additionally, from constraint (4) of MP, we can see that $1 - x_{i,b} - y_{i,b} - z_{i,b} \geq 0$ and $1 - y_{i,b} \geq 0$. Therefore, from (44) we can see that $\phi^*_{i+1,b} \geq 0$ as $\phi^*_{i,b} \geq 0$. Thus, by induction, we have proved $\phi^*_{i,b} \geq 0$ holds for $i \leq n$ and hence $\phi^*_{i,b}$ satisfies (40). Recall from (42), for $i = 1, \ldots, n$,

$$\begin{aligned}
\phi^*_{i,b} &= \phi^*_{i-1,b}\left(1 - y_{i-1,b}\right) + \left(t_i - t_{i-1}\right)\left(1 - x_{i-1,b} - y_{i-1,b} - z_{i-1,b}\right),\\
&= \phi^*_{i-1,b} - \phi^*_{i-1,b}y_{i-1,b} + t_i - t_{i-1} - \left(t_i - t_{i-1}\right)\left(x_{i-1,b} + y_{i-1,b} + z_{i-1,b}\right),\\
&\geq \phi^*_{i-1,b} - \Phi y_{i-1,b} + t_i - t_{i-1} - \Delta\left(x_{i-1,b} + y_{i-1,b} + z_{i-1,b}\right),
\end{aligned}$$

as $\phi^*_{i-1,b} \leq \Phi$ and $t_i - t_{i-1} \leq \Delta$, hence $\phi^*_{i,b}$ satisfies (37). Similarly for $i = 1, \ldots, n$,

$$\begin{aligned}
\phi^*_{i,b} &= \phi^*_{i-1,b}\left(1 - y_{i-1,b}\right) + \left(t_i - t_{i-1}\right)\left(1 - x_{i-1,b} - y_{i-1,b} - z_{i-1,b}\right),\\
&\geq \phi^*_{i-1,b} - \phi^*_{i-1,b}y_{i-1,b},\\
&\geq \phi^*_{i-1,b} - \Phi y_{i-1,b},
\end{aligned}$$

and therefore $\phi^*_{i,b}$ satisfies (38). As ϕ^*_b satisfies (36–40) it is a feasible solution to ϕ_b-SP, meaning the problem is feasible. □

Proposition 3.5 *θ_b-SP is feasible if and only if $\theta^*_{i,b} \leq \Theta$ for all $i \in \mathcal{I}$.*

Proof Both forward and reverse proofs follow analogously from the proofs of Proposition 3.4. □

We now show that in addition to determining the feasibility of the lifetime subproblems, θ^*_b and ϕ^*_b can outline periods of bank overruns, which can in turn be used to generate valid feasibility cuts.

Definition 3.6 (*Cleaning Overruns*) Let $\mathcal{X}_b$ denote the set of maintenance activities where bank $b \in \mathcal{B}$ was operational at the previous activity,

$$\mathcal{X}_b := \left\{ i \in \mathcal{I} : \begin{cases} x_{i-1,b} + y_{i-1,b} + z_{i-1,b} = 0, & \text{if } i \geq 1, \\ x_{i,b} + y_{i,b} + z_{i,b} = 0, & \text{if } i = 0, \end{cases} \right\}. \tag{45}$$

This gives the set of activities where the cleaning operational time is non-decreasing. Then, the set of overrun cleaning lifetimes is given as $\mathcal{C}_b$, where

$$\mathcal{C}_b := \left\{ (i, j) \in \mathcal{I} \times \mathcal{I} : \forall k \in [i, j], k \in \mathcal{X}_b, i - 1 \notin \mathcal{X}_b, \theta^*_{j,b} > \Theta \right\}. \tag{46}$$

If $(i, j) \in \mathcal{C}_b$, then from activity i to activity j, bank b is always operational and at activity j, the bank is overdue for a cleaning.

Definition 3.7 (Service Overruns) Let $\mathcal{Y}_b$ be the set of maintenance activities where bank $b \in \mathcal{B}$ was not serviced in the previous activity,

$$\mathcal{Y}_b := \left\{ i \in \mathcal{I} : \begin{cases} y_{i-1,b} = 0, & \text{if } i \geq 1, \\ y_{i,b} = 0, & \text{if } i = 0. \end{cases} \right\}. \tag{47}$$

This gives the set of activities where the service operational time is non-decreasing. The set of overrun service lifetimes is given as $\mathcal{V}_b$, where

$$\mathcal{V}_b := \left\{ (i, j) \in \mathcal{I} \times \mathcal{I} : \forall k \in [i, j], k \in \mathcal{Y}_b, i - 1 \notin \mathcal{Y}_b, \phi^*_{j,b} > \Phi \right\}. \tag{48}$$

If $(i, j) \in \mathcal{C}_b$, then from activity i to activity j, bank b is never serviced and at activity j, the bank is overdue for a service.

Definition 3.8 (*Standby Periods*) Let $\mathcal{O}_b$ be the set of activities where bank $b \in \mathcal{B}$ was either receiving a cleaning, or on standby at the previous activity,

$$\mathcal{O}_b := \left\{ i \in \{1, \ldots, n\} : x_{i-1,b} + z_{i-1,b} = 1 \right\}. \tag{49}$$

Proposition 3.9 *$\mathcal{V}_b = \emptyset$ if and only if ϕ_b-SP is feasible.*

Proof We first prove the forward statement. From constraint (7) of MP, there always exists an $i \in \mathcal{I}$ whereby $y_{i,b} = 0$, and therefore $\mathcal{Y}_b \neq \emptyset$. If $\mathcal{Y}_b \neq \emptyset$, then for any $j \in \{1, \ldots, n\}$ such that $\phi^*_{j,b} > 0$, we must have $y_{j-1,b} = 0$ and therefore from (47), $j \in \mathcal{Y}_b$. As $j \in \mathcal{Y}_b$, there always exists an $i \leq j$ such that $\forall k \in [i, j]$ we have $k \in \mathcal{Y}_b$ and $i - 1 \notin \mathcal{Y}_b$. However, if $\mathcal{V}_b = \emptyset$, then for all $j \in \mathcal{Y}_b$ such that $\phi^*_{j,b} > 0$, we must have $\phi^*_{j,b} \leq \Phi$, otherwise there exists an i such that $(i, j) \in \mathcal{V}_b$. Therefore $\phi^*_i \leq \Phi$ for all $i \in \mathcal{I}$, and hence from Proposition 3.4, ϕ_b-SP must be feasible.

The reverse statement is trivial. If ϕ_b-SP is feasible then from Proposition 3.4 we have $\phi^*_i \leq \Phi$ for all $i \in \mathcal{I}$, and hence there is no $j \in \mathcal{I}$ such that $\phi^*_{j,b} > \Phi$, therefore $\mathcal{V}_b = \emptyset$. □

Proposition 3.10 *$\mathcal{C}_b = \emptyset$ if and only if θ_b-SP is feasible.*

Proof Both forward and reverse proofs follow analogously from the proof of Proposition 3.9. □

Definition 3.11 (*Unbounded Direction*) Let $LP = \max \left\{ a^T x : Ax \leq 0, x \in \mathbb{R}^n \right\}$ be a linear program, where $a \in \mathbb{R}^n$ and $A \in \mathbb{R}^{m \times n}$. Then $v \in \mathbb{R}^n$ is an *unbounded direction* of LP if $Av \leq 0$ and $a^T v > 0$.

Proposition 3.12 *Fix a schedule $(x_{i,b}, y_{i,b}, z_{i,b}, t_i)$ for all $b \in \mathcal{B}$, $i \in \mathcal{I}$. Suppose further that ϕ_b-SP is infeasible with respect to the schedule $(x_{i,b}, y_{i,b}, z_{i,b}, t_i)$. Then, for all $(i, j) \in \mathcal{V}_b$, define $\pi^*, \rho^* \in \mathbb{R}^{n+1}$ as*

$$\pi_k^* = \begin{cases} 1, & \text{if } k \in [i, j] \text{ and } k \in \mathcal{X}_b, \\ 0, & \text{otherwise}, \end{cases} \qquad \rho_k^* = \begin{cases} -1, & \text{if } k = j, \\ 0, & \text{otherwise}, \end{cases}$$

for $k = 0, \ldots, n$, and define $\sigma^* \in \mathbb{R}^n$ as

$$\sigma_k^* = \begin{cases} 1, & \text{if } k \in [i, j] \text{ and } k \in \mathcal{O}_b, \\ 0, & \text{otherwise}, \end{cases}$$

for $k = 1, \ldots, n$. Then $(\pi^*, \sigma^*, \rho^*)$ is an unbounded direction of ϕ_b-DP.

Proof From Proposition 3.9, if ϕ_b-SP is infeasible with respect to the schedule $(x_{i,b}, y_{i,b}, z_{i,b}, t_i)$ then $\mathcal{V}_b$ is non-empty, and therefore the vector $(\pi^*, \sigma^*, \rho^*)$ exists. Moreover, the problem ϕ_b-DP is of the form $\max \left\{a^T x : Ax \leq 0, x \in \mathbb{R}^n\right\}$, and therefore matches that in Definition 3.11. We now prove $(\pi^*, \sigma^*, \rho^*)$ is feasible solution to ϕ_b-DP. The schedule generated by the master problem must satisfy constraint (4) and hence $x_{i,b} + y_{i,b} + z_{i,b} \leq 1$ for all $i \in \mathcal{I}$. Therefore from (45) and (49) we have that $\mathcal{X}_b \cap \mathcal{O}_b = \emptyset$ and hence for all $k \in [i, j]$ we have $\pi_k^* + \sigma_k^* = 1$. Therefore the only non-zero constraints are,

$$\begin{aligned} &\pi_k + \sigma_k - \pi_{k+1} - \sigma_{k+1} + \rho_k = 1 - 1 + 0 = 0 \leq 0, \quad k = i, \ldots, j-1, \\ &\pi_j + \sigma_j - \pi_{j+1} - \sigma_{j+1} + \rho_j = 1 - 0 - 1 = 0 \leq 0, \end{aligned}$$

which are all satisfied and hence $(\pi^*, \sigma^*, \rho^*)$ is feasible solution ϕ_b-DP.

The objective value Γ_b is given as,

$$\begin{aligned} \Gamma_b = \tilde{\phi}\pi_0^* &+ \sum_{k=1}^{n} \left(t_k - t_{k-1} - (\Phi + \Delta)\, y_{k-1,b} - \Delta \left(x_{k-1,b} + z_{k-1,b}\right)\right) \pi_k^* \\ &- \Phi \sum_{k=1}^{n} y_{k-1,b}\sigma_k^* + \Phi \sum_{k=0}^{n} \rho_k^*. \end{aligned} \tag{50}$$

From (47) we have that $0 \in \mathcal{Y}_b$ if and only if $1 \in \mathcal{Y}_b$. Therefore from (48), for all $(i, j) \in \mathcal{V}_b$, either $i = 0$ or $i \geq 2$. If $i \geq 2$, then (50) can be re-written with non-zero components,

$$\begin{aligned} \Gamma_b = &\sum_{\substack{k \in [i,j] \\ k \in \mathcal{X}_b}} \left(t_k - t_{k-1} - (\Phi + \Delta)\, y_{k-1,b} - \Delta \left(x_{k-1,b} + z_{k-1,b}\right)\right) \\ &- \Phi \sum_{\substack{k \in [i,j] \\ k \in \mathcal{O}_b}} y_{k-1,b} - \Phi. \end{aligned} \tag{51}$$

Now, from (49), for all $k \in \mathcal{O}_b$ we have $x_{k-1,b} + z_{k-1,b} = 1$ and hence from constraint (4) of MP we have that $y_{k-1,b} = 0$. Furthermore, if $x_{k-1,b} + z_{k-1,b} = 1$, then from (42), $\phi_{k,b}^* - \phi_{k-1,b}^* = 0 = y_{k-1,b}$. Additionally, from (45), for all $k \in \mathcal{X}_b$ we have $x_{k-1,b} = y_{k-1,b} = z_{k-1,b} = 0$ and therefore from (42) we have $\phi_k^* - \phi_{k-1}^* = t_k - t_{k-1}$. Then (51) can be simplified as follows,

$$\Gamma_b = \sum_{\substack{k \in [i,j] \\ k \in \mathcal{X}_b}} (t_k - t_{k-1}) + \sum_{\substack{k \in [i,j] \\ k \in \mathcal{O}_b}} \left(\phi_{k,b}^* - \phi_{k-1,b}^*\right) - \Phi,$$

$$= \sum_{\substack{k\in[i,j]\\ k\in\mathcal{X}_b}} \left(\phi^*_k - \phi^*_{k-1}\right) + \sum_{\substack{k\in[i,j]\\ k\in\mathcal{O}_b}} \left(\phi^*_{k,b} - \phi^*_{k-1,b}\right) - \Phi. \tag{52}$$

Given $\mathcal{X}_b \cap \mathcal{O}_b = \emptyset$ we can simplify (52) as follows,

$$\begin{aligned}\Gamma_b &= \sum_{k\in[i,j]} \left(\phi^*_k - \phi^*_{k-1}\right) - \Phi,\\ &= \phi^*_j - \phi^*_{i-1} - \Phi,\\ &= \phi^*_j - \Phi > 0,\end{aligned}$$

as $\phi^*_{i-1} = 0$ and $\phi^*_j > \Phi$. Therefore $(\pi^*, \sigma^*, \rho^*)$ is an unbounded direction of ϕ_b-DP. The proof for the case where $i = 0$ follows analogously. □

Proposition 3.13 *Fix a schedule $(x_{i,b}, y_{i,b}, z_{i,b}, t_i)$ for all $b \in \mathcal{B}$, $i \in \mathcal{I}$. Suppose further that θ_b-SP is infeasible with respect to the schedule $(x_{i,b}, y_{i,b}, z_{i,b}, t_i)$. Then, for all $(i, j) \in \mathcal{C}_b$, define $\gamma^*, \eta^* \in \mathbb{R}^{n+1}$ as*

$$\gamma^*_k = \begin{cases}1, & \text{if } k \in [i, j],\\ 0, & \text{otherwise},\end{cases} \qquad \eta^*_k = \begin{cases}-1, & \text{if } k = j,\\ 0, & \text{otherwise},\end{cases}$$

for $k = 0, \ldots, n$. Then (γ^*, η^*) is an unbounded direction of θ_b-DP.

Proof The proof follows analogously from the proof of Proposition 3.12. □

Proposition 3.14 *Fix a schedule $(x_{i,b}, y_{i,b}, z_{i,b}, t_i)$ for all $b \in \mathcal{B}$, $i \in \mathcal{I}$. For all banks $b \in \mathcal{B}$ where $\phi_b - SP$ is infeasible,*

$$\sum_{\substack{k\in[i,j]\\ k\in\mathcal{X}_b}} \left(t_k - t_{k-1} - (\Phi + \Delta)\, y_{k-1,b} - \Delta\left(x_{k-1,b} + z_{k-1,b}\right)\right) - \Phi \sum_{\substack{k\in[i,j]\\ k\in\mathcal{O}_b}} y_{k-1,b} \le \Phi, \quad \forall (i,j) \in \mathcal{V}_b : i \ge 2, \tag{53}$$

$$\tilde{\phi} + \sum_{\substack{k\in[1,j]\\ k\in\mathcal{X}_b}} \left(t_k - t_{k-1} - (\Phi + \Delta)\, y_{k-1,b} - \Delta\left(x_{k-1,b} + z_{k-1,b}\right)\right) - \Phi \sum_{\substack{k\in[1,j]\\ k\in\mathcal{O}_b}} y_{k-1,b} \le \Phi, \quad \forall (i,j) \in \mathcal{V}_b : i = 0, \tag{54}$$

are valid feasibility cuts for MP.

Proof These cuts are derived from the unbounded directions outlined in Proposition 3.12, and so from Benders (1962), they are valid cuts that remove this infeasible solution. Furthermore, the cuts only remove solutions that cause unbounded directions, and therefore infeasible subproblems. As such, the cuts do not remove other feasible solutions. Hence, they are valid feasibility cuts. □

Proposition 3.15 *Fix a schedule $(x_{i,b}, y_{i,b}, z_{i,b}, t_i)$ for all $b \in \mathcal{B}$, $i \in \mathcal{I}$. For all banks $b \in \mathcal{B}$ where $\theta_b - SP$ is infeasible,*

$$t_j - t_{i-1} - \sum_{k=i}^{j} \left((\Theta + \Delta)\left(x_{k-1,b} + y_{k-1,b}\right) + \Delta z_{k-1,b}\right) \le \Theta, \quad \forall (i,j) \in \mathcal{C}_b : i \ge 2,$$

$$t_j + \tilde{\theta}_b - \sum_{k=1}^{j} \left((\Theta + \Delta)\left(x_{k-1,b} + y_{k-1,b}\right) + \Delta z_{k-1,b}\right) \le \Theta, \quad \forall (i,j) \in \mathcal{C}_b : i = 0.$$

are valid feasibility cuts for MP.

Proof The proof follows analogously from the proof of Proposition 3.14. □

3.3 Valid inequalities

Entirely removing the operational due time constraints from the master problem has the potential to make its formulation weak. In such cases, it is common to introduce a set of valid inequalities to tighten the master problem. Doing so can reduce the number of feasibility cuts required, thereby leading to faster convergence. However, if the valid inequalities are weak then the model may perform worse due to the large number of unnecessary constraints included. Hence, having strong valid inequalities is highly desirable. We now formulate a set of valid inequalities based on practical assumptions of the problem, in hopes of tightening the master problem formulation.

The valid inequalities are based on the assumption that at any time, *exactly* one bank is offline in every subsystem. Consider the following formulation for the service due time for a single bank. Let intervals $i, j \in \mathcal{I}$ be such that $i < j$. Then the operational time for bank $b \in \mathcal{B}$ between intervals i and j is given as

$$\sum_{\substack{k=i \\ x_{k,b}+y_{k,b}+z_{k,b}=0}}^{j-1} \left(t_{k+1} - t_k\right). \tag{55}$$

In other words, it is the sum of the length of intervals between i and j where the bank is operational. Then, to ensure a bank is serviced on time, we must have

$$\sum_{\substack{k=i \\ x_{k,b}+y_{k,b}+z_{k,b}=0}}^{j-1} \left(t_{k+1} - t_k\right) \leq \Phi \left(1 + \sum_{k=i}^{j-1} y_{k,b}\right). \tag{56}$$

While this formulation ensures banks are serviced on time, the conditional sum makes it inappropriate for use in a mixed-integer linear model, and hence we proposed the formulation shown in Section 2.3. However, recall that a practical assumption of the model is that at any time, exactly one bank is offline in each subsystem. Therefore, if we sum (55) over all $b \in \mathcal{B}_s$ where $s \in \mathcal{S}$, we get

$$\sum_{b \in \mathcal{B}_s} \sum_{\substack{k=i \\ x_{k,b}+y_{k,b}+z_{k,b}=0}}^{j-1} \left(t_{k+1} - t_k\right) = \left(|\mathcal{B}_s| - 1\right)\left(t_j - t_i\right),$$

where $|\cdot|$ denotes the cardinality. Applying this to (56), we get the valid inequality

$$\left(|\mathcal{B}_s| - 1\right)\left(t_j - t_i\right) \leq \Phi \left(|\mathcal{B}_s| + \sum_{b \in \mathcal{B}_s} \sum_{k=i}^{j-1} y_{k,b}\right), \quad \forall s \in \mathcal{S}, i, j \in \mathcal{I} : i < j, \tag{57}$$

and similarly, for cleanings we get

$$\left(|\mathcal{B}_s| - 1\right)\left(t_j - t_i\right) \leq \Theta \left(|\mathcal{B}_s| + \sum_{b \in \mathcal{B}_s} \sum_{k=i}^{j-1} x_{k,b}\right), \quad \forall s \in \mathcal{S}, i, j \in \mathcal{I} : i < j. \tag{58}$$

These inequalities can then be added to the master problem to tighten its formulation.

4 Numerical results and discussion

We now explore the effectiveness of the proposed solution method using two case studies and several test instances. The case studies aim to provide the reader with a realistic problem setting by examining the scheduling requirements of two Bayer refineries based in Western Australia. The schedules generated in these case studies outline some common characteristics found in practical digester maintenance schedules. Using several test instances, we then assess the sensitivity of the model and solution method to various problem components, with the aim of identifying the factors that contribute to challenging instances. Specifically, we investigate how desired time horizon, service due time, and operational setup impact the model's performance. Furthermore, we analyse the contribution that Benders decomposition, valid inequalities, and lazy constraints make to overall algorithmic performance.

The scheduling model and solution algorithm was implemented in Gurobi version 10.0.1, using the *lazy constraint* callback feature. This feature allows the lazy constraints and Benders feasibility cuts to be integrated into the branch and cut framework. The program was run on a machine with a 2.3GHz AMD EPYC processor with 32 GB of RAM, using a single thread.

4.1 Alcoa case study

Alcoa of Australia operates two bauxite mines and three alumina refineries within Western Australia, producing a total of 9 million tonnes of alumina annually, making up approximately 7% of total production worldwide (Alcoa of Australia Limited, 2019). In this case study, we apply the digester scheduling model to the digestion setups in the Wagerup and Pinjarra refineries. The case study aims to provide readers with a realistic parameter selection and demonstrate some of the characteristics of a practical schedule.

In both case studies, the maintenance-related parameters are chosen as follows;

$$\begin{aligned} \alpha &= 35 \text{ days}, & \beta &= 80 \text{ days}, \\ \Theta &= 220 \text{ days}, & \Phi &= 680 \text{ days}, \\ A &= \$10, & B &= \$40. \end{aligned}$$

The cleaning overlap cost L was set at 10% the cost per day of a cleaning activity, and the service overlap cost M was set at 25% the cost per day of a service activity. Each bank's starting operational time since its last cleaning, $\tilde{\theta}_b$ is chosen randomly such that $\tilde{\theta}_b \in [0, \Theta]$. Similarly, each bank's starting operational time since its last service, $\tilde{\phi}_b$ is chosen randomly such that $\tilde{\phi}_b \in [0, \Phi]$. A practical schedule should last for approximately three years. During this time, there are expected to be two valve changes for each production unit.

4.1.1 Alcoa Wagerup

Alcoa Wagerup uses three digester banks set up in a single subsystem to complete the digestion phase of the Bayer process. Figure 3 outlines an example of this setup. For this setup $\mathcal{S} = \{1\}$ and $\mathcal{B} = \{1, 2, 3\}$. A double bank switch occurs when bank 1 is switched with bank 3, and vice versa, hence $\mathcal{D}_1 = \{(1, 3), (3, 1)\}$.

Table 2 summarizes the performance of the original model and the decomposition approach proposed in Section 3 on the Wagerup case study. The original model was solved in 14 seconds using Gurobi. The use of Benders decomposition and valid inequalities resulted in a nearly 50% reduction in solve time, indicating a substantial improvement due to the application of Benders decomposition.

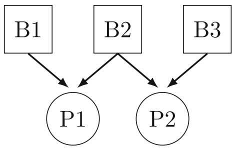

Fig. 3 Wagerup digestion system with three banks feeding two production units

Table 2 Performance of the original model and Benders decomposition solution algorithm on the Wagerup and Pinjarra case studies

Case study	Original			Decomposed		
	Time (sec)	Gap (%)	Objective value	Time (sec)	Gap (%)	Objective Value
Wagerup	14.00	0.00	240.0	7.46	0.00	240.0
Pinjarra	7200.02	32.30	1011.0	7200.02	7.52	1010.0

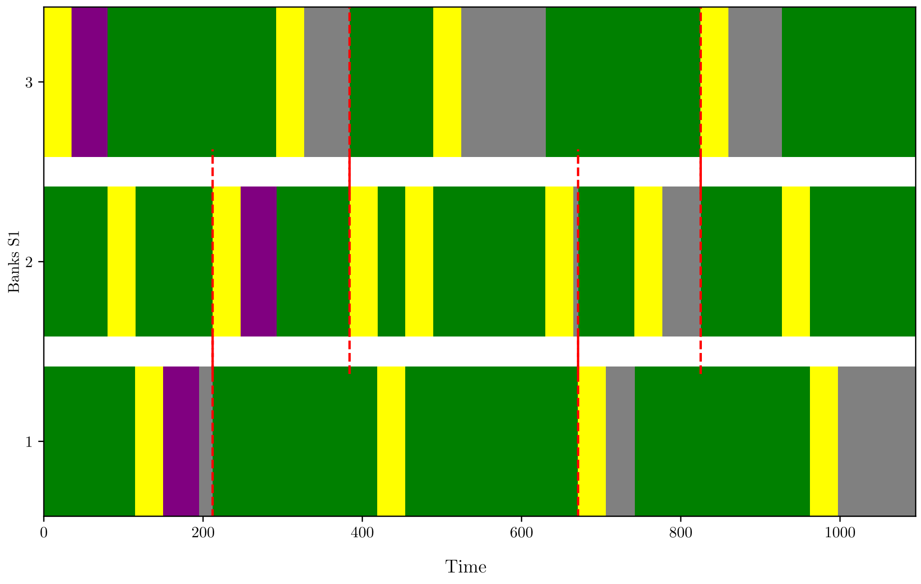

Fig. 4 Optimal three-year maintenance schedule for Wagerup case study. Green represents operational periods, yellow represents cleaning activities, purple represents service activities and grey represents standby time. Valve changes are shown as red dashed lines that cross over their associated banks. For instance, the valve change on day 212 occurs on production unit 1, as it can connect to banks 1 and 2

Figure 4 displays the optimal three-year maintenance schedule for the Wagerup case study, which includes fifteen cleanings and three services while meeting all four planned valve change days. Notably, to prevent double bank changes, the schedule frequently cleans bank 2. Consequently, the average duration of an operational period for banks 1 and 3 is significantly longer than for bank 2. This is a common characteristic of digester maintenance schedules.

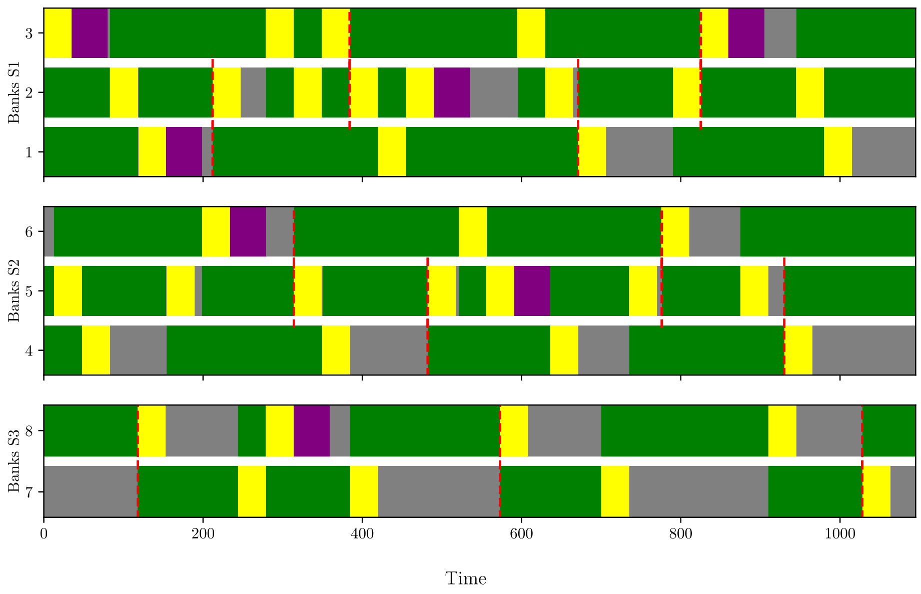

Fig. 5 Best known three-year maintenance schedule for the Pinjarra case study

4.1.2 Alcoa Pinjarra

The Alcoa Pinjarra refinery contains eight digester banks split into three subsystems. Figure 1 outlines an example of this setup. For this setup, $\mathcal{S} = \{1, 2, 3\}$ with $\mathcal{B}_1 = \{1, 2, 3\}$, $\mathcal{B}_2 = \{4, 5, 6\}$ and $\mathcal{B}_3 = \{7, 8\}$. Double bank switching occurs when bank 1 is switched with bank 3 or bank 4 is switched with bank 6 and vice versa, hence $\mathcal{D}_1 = \{(1, 3), (3, 1)\}$, $\mathcal{D}_2 = \{(4, 6), (6, 4)\}$ and $\mathcal{D}_3 = \emptyset$.

Table 2 presents the performance of the Benders decomposition approach and the original model for the Pinjarra case study model. Apart from Benders decomposition and valid inequalities, a lazy constraint formulation of constraint (15) can be utilized, given the presence of multiple subsystems in the Pinjarra setup. Despite two hours of solve time, neither the original model nor Benders decomposition could attain the optimal solution, indicating that the larger and more complex operational setup poses a significantly more challenging problem. Nevertheless, Benders decomposition yields an improved objective value and considerably reduces the optimality gap.

Figure 5 displays the best-known three-year maintenance schedule for the Pinjarra setup, which includes a total of 39 cleanings, 7 services and meets all ten planned valve changes. To avoid double bank changes, the schedule frequently plans cleanings for banks 2 and 5. Moreover, we observe that no more than two cleanings or services occur concurrently, and efforts are made to minimize any potential clashes.

4.2 Test instances

The analysis in the previous case study was limited to only two problem settings and only compared the proposed solution algorithm from Section 3 with the original model. To provide a more comprehensive understanding of the model's sensitivities and complexities, we introduce several test instances. These instances aim to explore how time horizon, service due time and operational setups impact model performance. Additionally, we investigate the contribution that Benders decomposition, valid inequalities, and lazy constraints make to overall algorithmic performance and attempt to determine which approach better tackles the model complexities.

4.2.1 Time horizon

To explore the effect time horizon has on model performance, we solve the Wagerup and Pinjarra case study models with varying time horizons. Furthermore, we use different combinations of Benders decomposition, valid inequalities, and lazy constraints to gain insights into the most effective technique for handling increasing time horizons.

Table 3 outlines the performance of the different solution strategies for the Wagerup scheduling model with time horizons ranging from two to five years. The results show that all solution approaches are highly sensitive to increasing time horizons. While all approaches solved the two-year schedule in under a second, the solve time increased exponentially with an increase in time horizon. No solver was able to prove optimality within two hours of solve time for a five-year schedule. For the $\tau = 730$ and $\tau = 1460$, the original model solved the fastest. However, for $\tau = 1095$, Benders decomposition with lazy constraints solved in half the time of the original model. Interestingly, for $\tau = 1095$ and $\tau = 1460$, the introduction of valid inequalities appeared to slow the original model, indicating they may have been ineffective and weighed down the solver. In contrast, valid inequalities improved the performance of Benders decomposition substantially. For instance, for the case where $\tau = 1095$, valid inequalities reduced the number of service cuts by 50%, leading to a far better runtime.

Table 4 outlines the performance of the different solution strategies for the Pinjarra scheduling model with time horizons ranging from two to four years. As this problem is more difficult than the Wagerup problem, it is more sensitive to increasing time horizons, with no approach able to solve the three-year schedule within a two-hour time limit. For $\tau = 730$, the best performance was achieved by Benders decomposition with lazy constraints, solving in 106 seconds. Additionally, the use of lazy constraints appears very effective in this case. For the four-year schedule, Benders decomposition was not able to find an integer feasible solution after two hours. Remarkably, Benders decomposition with valid inequalities achieved a far tighter best bound than any other approach, despite not finding an integer feasible solution. Finally, valid inequalities were highly effective for the larger time horizon, significantly reducing the number of added service and cleaning cuts and tightening the best bound.

Overall, we can conclude that the scheduling model is highly sensitive to the time horizon, with all solution approaches struggling to solve longer schedules within a reasonable time limit. However, the use of valid inequalities and Benders decomposition leads to far tighter best bounds at large time horizons, although these approaches can struggle to find good quality feasible solutions. Furthermore, the use of lazy constraints appears very effective when combined with Benders decomposition.

4.2.2 Service due time

The case study highlighted the challenges associated with planning services, as operational periods can vary significantly in length, making it difficult to determine when to service a bank. To better understand the effect of this complexity, we explore how different service

Table 3 Performance of various solution approaches for the Wagerup case study for varying time horizons (two to five years) with a two-hour time limit. We report the solve time in seconds, optimality gap, objective value and best bound for each approach. For Benders decomposition, we also report the number of cleaning and service cuts added

τ	Benders decomposition	Valid inequalities	Time (sec)	Gap (%)	Objective value	Best bound	Cleaning cuts	Service cuts
730	✓	✓	0.93	0.00	170.0	170.0	61	217
730	✓		0.89	0.00	170.0	170.0	76	212
730		✓	0.61	0.00	170.0	170.0		
730			0.50	0.00	170.0	170.0		
1095	✓	✓	7.46	0.00	240.0	240.0	92	570
1095	✓		19.25	0.00	240.0	240.0	93	1068
1095		✓	21.58	0.00	240.0	240.0		
1095			14.00	0.00	240.0	240.0		
1460	✓	✓	324.80	0.00	340.0	340.0	151	6122
1460	✓		325.66	0.00	340.0	340.0	167	8312
1460		✓	392.17	0.00	340.0	340.0		
1460			276.95	0.00	340.0	340.0		
1825	✓	✓	7200.04	6.98	430.0	400.0	159	34362
1825	✓		7200.11	13.95	430.0	370.0	240	56812
1825		✓	7200.01	9.30	430.0	390.0		
1825			7200.01	11.63	430.0	380.0		

Table 4 Performance of various solution approaches for the Pinjarra case study for varying time horizons (two to four years) with a two-hour time limit. We report the solve time in seconds, optimality gap, objective value, best bound and number of each type of cut added

τ	Benders decomposition	Valid inequalities	Lazy constraints	Time (sec)	Gap (%)	Objective value	Best bound	Cleaning cuts	Service cuts	Lazy added
730	✓	✓	✓	142.73	0.00	690.0	690.0	560	2524	375
730	✓	✓		209.58	0.00	690.0	690.0	598	2515	
730	✓		✓	105.90	0.00	690.0	690.0	772	2543	651
730	✓			190.70	0.00	690.0	690.0	604	2738	
730		✓		850.65	0.00	690.0	690.0			
730			✓	128.84	0.00	690.0	690.0			90
730				310.39	0.00	690.0	690.0			
1095	✓	✓	✓	7200.02	7.52	1010.0	934.0	979	16965	47
1095	✓	✓		7200.06	10.81	1010.0	901.0	1299	17800	
1095	✓		✓	7200.01	20.42	1087.0	865.0	1321	32947	63
1095	✓			7200.03	19.33	1080.0	871.0	1311	27691	
1095		✓		7200.05	34.99	1011.0	657.0			
1095			✓	7200.00	24.58	1011.0	763.0			290
1095				7200.02	32.30	1011.0	684.0			
1460	✓	✓	✓	7200.01	–	–	1088.0	952	29158	85
1460	✓	✓		7200.05	–	–	1057.0	953	25492	
1460	✓		✓	7200.05	–	–	775.0	1616	79313	64
1460	✓			7200.13	–	–	834.0	1707	69396	
1460		✓		7200.06	39.73	1407.0	848.0			
1460			✓	7200.01	43.10	1385.0	788.0			16
1460				7200.06	60.88	1568.0	613.0			

due times impact model performance. Using the Wagerup operational setup over a three-year time horizon, we set $\Phi = p\Theta$ where $p = 0, 2, 3, 4$. For each value of p, we then examine the performance of the various solution approaches. Note that when $p = 0$, we assume no servicing is required.

Table 5 summarizes the performance of various solution approaches for each value of p. The findings indicate that the performance of the different solution strategies varies significantly depending on the service due time. For example, when no servicing is required ($p = 0$), the model is easily solved in under 10 seconds, with Benders decomposition proving particularly effective, solving the model to optimality in less than a second. However, when $\Phi = 2\Theta$, Benders decomposition becomes vastly ineffective, achieving only a 9.76% gap after two hours of solve time, whereas the original model solved to optimality in under 400 seconds. On the other hand, when $\Phi = 3\Theta$, the proposed method of Benders decomposition and valid inequalities was the best performer, improving on the original model by 400 seconds. The use of valid inequalities appears very effective in this case, reducing the number of service cuts by almost half. However, when increasing to $\Phi = 4\Theta$, Benders decomposition once again performs worse, and when used without valid inequalities, it is not able to prove optimality in two hours. Finally, when $\Phi = 5\Theta$, the optimal solution contained no services across the three-year time horizon. Nevertheless, Benders decomposition was able to prove optimality far quicker, requiring only 22 service lifetime cuts when used with valid inequalities.

Service due time represents a complicated and sensitive part of the model. The performances of all solution algorithms vary dramatically when service due time changes with respect to the cleaning due time. Therefore, the decision of which solution algorithm to use should depend on the ratio. Further analysis and research is necessary to better understand this trend and find ways of overcoming this sensitivity.

4.2.3 Operational setup

The previous experiments were limited to the operational setups introduced in the case study. However, upon comparing the results presented in Tables 3 and 4, it is apparent that there is a significant difference in problem complexity between the Wagerup and Pinjarra setups. To further understand the impact of varying the size and layout of the operational setup, we introduce several new test instances. Specifically, for each instance, we assume there are m subsystems, and within each subsystem, there are n banks, resulting in a total of mn banks. We set up the banks such that there are no double bank changes and assume no planned valve change days. All other maintenance-related parameters are identical to those in the Pinjarra case study. The problem is then solved over a two-year time horizon.

The performance of the various solution approaches on different operational setups are presented in Table 6. Interestingly, in all examples where there were two banks per subsystem, the decomposition approach yielded significantly better results. For instance, when two banks were spread across four subsystems, Benders decomposition with valid inequalities achieved optimality in 465 seconds, while the original model only achieved a 75% gap after two hours of solve time. Moreover, including more banks within the same subsystem creates a much more difficult problem. For example, the problem of two banks per subsystem spread across four subsystems (totalling eight banks) was easily solved in under 500 seconds. However, if the eight banks were split across only two subsystems, the performance was far worse, with the best-performing model achieving only a 36% gap after two hours of solve time. Furthermore, for these large and very challenging models, the decomposition approach with valid inequalities consistently outperformed in terms of both objective value and best bound.

Table 5 Performance of various solution approaches for varying service due times where $\Phi = p\Theta$. For each Φ and solution approach, we report the solve time in seconds, optimality gap, objective value, best bound and number of each type of cut added

p	Benders decomposition	Valid inequalities	Time (sec)	Gap (%)	Objective value	Best bound	Cleaning cuts	Service cuts
0	✓	✓	1.90	0.00	190.0	190.0	136	
0	✓		0.60	0.00	190.0	190.0	153	
0		✓	9.76	0.00	190.0	190.0		
0			5.66	0.00	190.0	190.0		
2	✓	✓	7200.01	9.76	410.0	370.0	121	22691
2	✓		7200.01	24.39	410.0	310.0	176	32540
2		✓	1100.55	0.00	410.0	410.0		
2			387.05	0.00	410.0	410.0		
3	✓	✓	315.90	0.00	280.0	280.0	157	7082
3	✓		1964.08	0.00	280.0	280.0	172	13007
3		✓	388.61	0.00	280.0	280.0		
3			716.07	0.00	280.0	280.0		
4	✓	✓	2399.56	0.00	260.0	260.0	151	10049
4	✓		7200.01	3.85	260.0	250.0	174	23275
4		✓	1802.64	0.00	260.0	260.0		
4			1159.70	0.00	260.0	260.0		
5	✓	✓	1.89	0.00	190.0	190.0	120	22
5	✓		0.66	0.00	190.0	190.0	156	83
5		✓	36.60	0.00	190.0	190.0		
5			16.66	0.00	190.0	190.0		

Table 6 Performance of various solution approaches for varying operational setup. Each setup consists of n banks per subsystem, with m subsystems. For each setup and solution approach, we report the solve time in seconds, optimality gap, objective value, best bound and number of each type of cut added

n	m	Benders decomposition	Valid inequalities	Lazy constraints	Time (sec)	Gap (%)	Objective value	Best bound	Cleaning cuts	Service cuts	Lazy added
2	2	✓	✓	✓	9.67	0.00	60.0	60.0	562	2	0
2	2	✓	✓		15.04	0.00	60.0	60.0	583		
2	2	✓			47.32	0.00	60.0	60.0	2371	22	
2	2		✓		5690.25	0.01	60.0	60.0			
2	2			✓	2111.90	0.00	60.0	60.0			0
2	2				2896.40	0.01	60.0	60.0			
2	3	✓	✓	✓	76.14	0.00	90.0	90.0	1000	1	0
2	3	✓	✓		97.12	0.00	90.0	90.0	754		
2	3	✓			161.85	0.00	90.0	90.0	3653	39	
2	3		✓		7200.01	35.66	90.0	58.0			
2	3			✓	7200.00	52.52	90.0	43.0			0
2	3				7200.01	57.64	90.0	38.0			
2	4	✓	✓	✓	678.22	0.00	120.0	120.0	1277	1	0
2	4	✓	✓		465.63	0.00	120.0	120.0	1112	2	
2	4	✓			1931.50	0.01	120.0	120.0	4759	48	
2	4		✓		7200.01	52.40	120.0	57.0			
2	4			✓	7200.00	69.18	120.0	37.0			0
2	4				7200.01	73.18	120.0	32.0			
3	2	✓	✓	✓	7200.01	25.30	120.0	90.0	1701	10	0
3	2	✓	✓		7200.01	21.96	120.0	94.0	1486	10	

Table 6 continued

n	m	Benders decomposition	Valid inequalities	Lazy constraints	Time (sec)	Gap (%)	Objective value	Best bound	Cleaning cuts	Service cuts	Lazy added
3	2	✓			7200.01	24.21	120.0	91.0	4006	40	
3	2		✓		7200.01	39.39	120.0	73.0			
3	2			✓	7200.00	65.92	120.0	41.0			0
3	2				7200.01	71.33	120.0	34.0			
3	3	✓	✓	✓	7200.01	41.46	205.0	120.0	2351	9	0
3	3	✓	✓		7200.02	36.62	190.0	120.0	1923	9	
3	3	✓			7200.01	44.43	205.0	114.0	6012	60	
3	3		✓		7200.01	46.78	205.0	109.0			
3	3			✓	7200.00	85.92	235.0	33.0			0
3	3				7200.01	86.26	205.0	28.0			
4	2	✓	✓	✓	7200.01	36.84	190.0	120.0	2574	20	0
4	2	✓	✓		7200.03	41.46	205.0	120.0	2954	20	
4	2	✓			7200.02	43.99	205.0	115.0	5589	56	
4	2		✓		7200.02	41.91	205.0	119.0			
4	2			✓	7200.00	82.05	205.0	37.0			0
4	2				7200.01	84.17	205.0	32.0			
4	3	✓	✓	✓	7200.04	68.14	565.0	180.0	4390	29	0
4	3	✓	✓		7200.02	69.49	590.0	180.0	4462	28	
4	3	✓			7200.01	74.86	590.0	148.0	8369	82	
4	3		✓		7200.02	69.72	590.0	179.0			
4	3			✓	7200.00	94.21	580.0	34.0			5
4	3				7200.01	95.31	600.0	28.0			

5 Conclusion

In this paper, we formulated a maintenance scheduling model for digester banks, a critical asset used in the Bayer process. Our research was motivated by the importance digestion plays in the Bayer process and the difficulty of determining cost-efficient maintenance schedules for fleets of digester banks. Due to the network nature of digestion systems and complex maintenance requirements, scheduling bank maintenance manually can be challenging. Therefore, we propose a scheduling model that can find the cost-optimized maintenance schedule that satisfies all required constraints. While this research focuses on Bayer digestion, many maintenance scheduling problems in refinery settings exhibit similar challenges.

Several strategies were introduced to assist in solving the problem at larger dimensions. Benders decomposition was used to handel the complicated operational lifetime requirement of the banks. We showed how the Benders subproblems could be solved easily using a specialist algorithm. Additionally, valid inequalities based on practical assumptions were introduced to further tighten the master problem. Finally, lazy constraints were used to handel service clash constraints, which make up a substantial proportion of the constraint set, yet only a small proportion are ever active. These strategies were then evaluated on two case studies involving real world digester setups. Several test instances were also generated to further explore the effectiveness of each strategy as well as better understand the key complexities of the problem.

Extensive numerical experiments highlight the key model sensitives and complexities. Parameters such as time horizon, service due time and operational setup each have significant impacts on the performance of the suggested solution strategies. Crucially, the results show that no single strategy is better in all cases. The model's sensitives mean that particular strategies perform better in particular settings.

As such, a compelling avenue for future research is to better understand each of these complexities and find ways to overcome them. In particular, the model should become more stable to increasing time horizons. This is partially taken care of by the continuous-time model, however, results showed that the desired time horizon still limits the model. Moreover, numerical experiments show how the number of banks within a subsystem impact the performance of the model. This should be explored further to assist in solving more general digestion setups. Making such improvements would make the model more useful in a practical setting.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions This work is supported by the Australian Research Council through the Centre for Transforming Maintenance through Data Science (grant number IC180100030), funded by the Australian Government.
The authors acknowledge the support of Alcoa of Australia for their assistance in formulating this research problem and for making available the data utilised in this manuscript. This work is supported by resources provided by the Pawsey Supercomputing Research Centre with funding from the Australian Government and the Government of Western Australia.

Declarations

Conflict of interest The authors declare no conflicts of interest regarding the research presented in this article.

References

Alcoa of Australia Limited. (2019). 2019 Tax Transparency Report. Retrieved from https://www.alcoa.com/australia/en/pdf/alcoa-of-australia-ltdtax-transparency-report-2019.pdf

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik, 4*(1), 238–252. https://doi.org/10.1007/BF01386316

Cheng, L.-W., Wang, Y.-L., Zhou, Q.-S., Qi, T.-G., Liu, G.-H., Peng, Z.-H., & Li, X.-B. (2021). Scale Formation During the Bayer Process and a Potential Prevention Strategy. *Journal of Sustainable Metallurgy, 7*(3), 1293–1303. https://doi.org/10.1007/S40831-021-00417-4

Contreras, I., Cordeau, J. F., & Laporte, G. (2011). Benders Decomposition for Large-Scale Uncapacitated Hub Location. *Operations Research, 59*(6), 1477–1490. https://doi.org/10.1287/OPRE.1110.0965

de Jonge, B., & Scarf, P. A. (2020). A review on maintenance optimization. *European Journal of Operational Research, 285*(3), 805–824. https://doi.org/10.1016/j.ejor.2019.09.047

Fischetti, M., Ljubic, I., & Sinnl, M. (2016). Redesigning Benders Decomposition for Large-Scale Facility Location. *Management Science, 63*(7), 2146–2162. https://doi.org/10.1287/MNSC.2016.2461

Floudas, C. A., & Lin, X. (2004). Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review. *Computers & Chemical Engineering, 28*(11), 2109–2129. https://doi.org/10.1016/J.COMPCHEMENG.2004.05.002

Heydar, M., Mardaneh, E., & Loxton, R. (2021). Approximate dynamic programming for an energy-efficient parallel machine scheduling problem. *European Journal of Operational Research*. https://doi.org/10.1016/J.EJOR.2021.12.041

Hooker, J. N. (2007). Planning and Scheduling by Logic-Based Benders Decomposition. *Operations Research, 55*(3), 588–602. https://doi.org/10.1287/OPRE.1060.0371

Koné, O., Artigues, C., Lopez, P., & Mongeau, M. (2011). Event-based MILP models for resource-constrained project scheduling problems. *Computers & Operations Research, 38*, 3–13. https://doi.org/10.1016/j.cor.2009.12.011

Kopanos, G. M., Kyriakidis, T. S., & Georgiadis, M. C. (2014). New continuoustime and discrete-time mathematical formulation for resource-constrained project scheduling problems. *Computers and Chemical Engineering, 68*, 96–106. https://doi.org/10.1016/J.COMPCHEMENG.2014.05.009

Lerch, D., & Trautmann, N. (2019). A Lazy-Constraints Approach to Resource- Constrained Project Scheduling. *IEEE International Conference on Industrial Engineering and Engineering Management, 144–148*,. https://doi.org/10.1109/IEEM44572.2019.8978524

Li, X., Yu, S., Dong, W., Chen, Y., Zhou, Q., Qi, T., & Jiang, Y. (2015). feb). Investigating the effect of ferrous ion on the digestion of diasporic bauxite in the Bayer process. *Hydrometallurgy, 152*, 183–189. https://doi.org/10.1016/J.HYDROMET.2015.01.001

Manne, A. S. (1960). On the Job-Shop Scheduling Problem. *Operations Research, 8*(2), 219–223. https://doi.org/10.1287/OPRE.8.2.219

Maravelias, C. T., & Grossmann, I. E. (2003). New General Continuous-Time State-Task Network Formulation for Short-Term Scheduling of Multipurpose Batch Plants. *Industrial and Engineering Chemistry Research, 42*(13), 3056–3074. https://doi.org/10.1021/IE020923Y

Miller, C. E., Tucker, A. W., & Zemlin, R. A. (1960). Integer Programming Formulation of Traveling Salesman Problems. *Journal of the ACM, 7*, 326–329. https://doi.org/10.1145/321043.321046

Naoum-Sawaya, J., & Buchheim, C. (2016). Robust critical node selection by benders decomposition. *INFORMS Journal on Computing, 28*(1), 162–174. https://doi.org/10.1287/ijoc.2015.0671

Olde Keizer, M. C., Teunter, R. H., & Veldman, J. (2016). Clustering condition-based maintenance for systems with redundancy and economic dependencies. *European Journal of Operational Research, 251*(2), 531–540. https://doi.org/10.1016/J.EJOR.2015.11.008

Pearce, R. H. (2019). Towards a general formulation of lazy constraints (PhD Thesis). School of Mathematics and Physics, The University of Queensland.

Pearce, R. H., & Forbes, M. (2018). Disaggregated benders decomposition for solving a network maintenance scheduling problem. *Journal of the Operational Research Society, 70*(6), 941–953. https://doi.org/10.1080/01605682.2018.1471374

Rahmaniani, R., Crainic, T. G., Gendreau, M., & Rei, W. (2017). The Benders decomposition algorithm: A literature review. *European Journal of Operational Research, 259*(3), 801–817. https://doi.org/10.1016/J.EJOR.2016.12.005

Seif, Z., Mardaneh, E., Loxton, R., & Lockwood, A. (2020). Minimizing equipment shutdowns in oil and gas campaign maintenance. *Journal of the Operations Research Society*. https://doi.org/10.1080/01605682.2020.1745699

Siopa, J. M. P., Garção, J. E. S., & Silva, E. J. M. (2015). Component redundancy allocation in optimal cost preventive maintenance scheduling. *Journal of the Operational Research Society, 66*(6), 925–935.