

A unified modelling framework and improved formulations for single-hoist cyclic scheduling

Markó Horváth 

HUN-REN Institute for Computer Science and Control, Kende u. 13-17., Budapest, 1111, Hungary

ARTICLE INFO

Keywords:

Cyclic hoist scheduling
Mathematical modelling
Mixed-integer linear programming
Constraint programming

ABSTRACT

The cyclic hoist scheduling problem originates in electroplating lines, where a single or multiple hoists transport parts between processing tanks subject to technological constraints. The objective is typically to determine a cyclic sequence of hoist movements that minimizes the cycle time while satisfying travel and processing constraints. Although the problem has been widely studied for several decades, the literature contains a puzzling phenomenon: different studies often report different optimal cycle times for the same benchmark instances, which limits the comparability and reproducibility of computational results.

In this paper, we revisit the modelling of cyclic hoist scheduling problems from a unified perspective. We introduce a consistent modelling approach for single-hoist problems and analyse several mixed-integer linear programming (MIP) formulations proposed in the literature. Our analysis identifies modelling inconsistencies and clarifies the relationships between existing formulations. Based on these observations, we propose natural constraint programming (CP) models that can serve as baseline approaches, and we also derive improved MIP formulations. Extensive computational experiments compare the strength and performance of the investigated formulations.

To support reproducible research, we also provide a publicly available library containing benchmark instances and implementations of several CP and MIP formulations for single-hoist cyclic hoist scheduling.

1. Introduction

The *hoist scheduling problem* (HSP) originates in electroplating facilities, where hoists transfer parts between chemical baths according to their predefined processing sequences. The problem also naturally arises in other material handling contexts, such as robotic cell scheduling (e.g., Dawande et al. (2005) and Feng et al. (2024)). Briefly stated, a single or multiple hoists transfer carriers containing the parts between tanks, and the scheduling of these movements must satisfy several technological constraints, including minimum and maximum processing times. The *cyclic hoist scheduling problem* (CHSP) aims to determine a repetitive sequence of hoist movements, called a *cycle*, which is then executed continuously in order to process a large number of identical parts. Determining a cycle with minimal duration is therefore directly related to maximizing the throughput of the production line.

50 years ago, Phillips and Unger (1976) published their paper in which the authors considered a problem to minimize the cycle time for a single hoist. Since then, several solution approaches have been developed for the problem and its variants, including the multi-hoist case. These approaches include mixed-integer linear programming (e.g., Li and Fung (2014), Nait-Sidi-Moh and El-Amraoui (2016), Feng et al. (2018) and Mao et al. (2018)), constraint programming (e.g., Wallace

and Yorke-Smith (2020) and Liu et al. (2025)), other exact approaches (e.g., Riera and Yorke-Smith (2002), Che and Chu (2004), Leung and Levner (2006) and Lei et al. (2014)), and heuristics or metaheuristic algorithms (e.g., Chtourou et al. (2013), El Amraoui and Elhafsi (2016), Yan et al. (2016), Laajili et al. (2021) and El Amraoui and Benrejeb (2024)).

Despite the extensive literature, the examination of published results reveals a puzzling phenomenon. Different studies that are supposed to solve the same basic problem often report different optimal cycle times for the same benchmark instances. Table 1 illustrates this issue for a basic single-hoist problem for several widely used instances, where the reported optimal values vary significantly across papers. Even in the most recent articles, we can find results that are completely different from all previous ones.

Such discrepancies significantly limit the comparability of different approaches. In many cases, new methods are evaluated primarily by comparing their results with previously reported cycle times. If these reference values are inconsistent or incorrect, it becomes difficult to assess the true performance of new algorithms. This issue is particularly problematic for heuristic methods, whose quality may be measured to incorrect reference values.

E-mail address: marko.horvath@sztaki.hu.

<https://doi.org/10.1016/j.cor.2026.107600>

Received 27 March 2026; Received in revised form 12 June 2026; Accepted 30 June 2026

Available online 6 July 2026

0305-0548/© 2026 The Author. Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

Table 1

Reported values in the literature for a basic single-hoist problem on some benchmark instances.

Paper	Instances							
	PU	Mini	BO1	BO2	Cu	Zn	Ligne 1	Ligne 2
Phillips and Unger (1976)	580	–	–	–	–	–	–	–
Armstrong et al. (1994)	521	–	304.1	255.7	319.95	435.85	–	–
Leung et al. (2004)	521	281	299.5	279.3	1847.2	1743.4	–	–
Che et al. (2011)	521	–	–	–	–	–	393	712
Che et al. (2015)	521	287	281.9	279.3	–	–	392	712
Mao et al. (2018)	–	284	–	279.3	–	1743.4	–	–
Laajili et al. (2021) ^a	521	340	333.2	279.3	1847.2	1743.4	425	712
Liu et al. (2025)	518	284	267	257	1847	1742	392	722

– not reported.

^a Values are not indicated as optimal.

Importantly, these inconsistencies rarely arise due to fundamentally different problem definitions. Hoist scheduling problems have been systematically classified by Manier and Bloch (2003), who introduced a unified notation, similar to the well-known classification scheme of Graham et al. (1979) for machine scheduling problems. This notation allows researchers to precisely describe the main characteristics of the investigated hoist scheduling problem. In principle, this should make it possible to define and reproduce the same problem settings across studies.

Although the notation used in the literature is relatively unified, the formalization of the problems is much less consistent. In many studies, the constraints are described only in text, without providing an explicit mathematical formulation that clearly characterizes the set of feasible solutions. To investigate the sources of these inconsistencies, we therefore focus primarily on constraint programming (CP) and mixed-integer linear programming (MIP) models. In these approaches, the constraints must be explicitly specified in mathematical form, which makes it possible to clearly understand how the authors interpret the technological restrictions of the problem. Thus, these formulations provide a transparent and reproducible basis for analysing and comparing different models. Our analysis of the literature suggests that several factors may contribute to the observed discrepancies. In some cases, optimality claims appear to be incorrect; in others, published formulations contain modelling inaccuracies or omit important constraints. Subtle differences in modelling assumptions may also lead to different feasible solution spaces, and benchmark instances themselves may be used in slightly different forms. As a result, formulations that are intended to represent the same problem may in fact correspond to different variants.

These observations highlight the need for a systematic and reproducible modelling framework for cyclic hoist scheduling problems. In this paper, we therefore revisit the modelling of cyclic hoist scheduling problems from a unified perspective. Our main contributions are as follows: (1a) a unified modelling approach for single-hoist cyclic scheduling; (1b) new constraint programming formulations; (2a) revision of existing mixed-integer linear programming formulations from the literature; (2b) strengthened MIP formulations; (3) computational comparison of MIP formulations; and (4) a public library containing benchmark instances and solution approaches.

Unified modelling of single-hoist problems, new CP formulations. We propose a unified modelling framework for single-hoist cyclic scheduling problems, in which the requirements of feasible cyclic schedules are systematically identified, analysed, and formalized within a common notation and modelling structure. The fundamental requirements considered in the framework include travel-time feasibility, soaking constraints, and load-unload configurations; and the framework also covers several important extensions, including multi-function tanks, multi-tank (multi-stage) operations, multi-degree problems (multi-cycles), and multi-part problems. Compared to existing models in the literature, our approach provides a more systematic treatment of extensions such as multi-function tanks and multi-degree problems, and explicitly

accounts for load-unload configurations, which are often neglected in previous formulations. In particular, the resulting formulation for multi-cycles arises naturally from the simple cycle model and is more transparent than existing formulations. The framework also provides a foundation for future extensions to multi-hoist problems. An additional consequence of this modelling framework is that the resulting scheduling requirements naturally translate into constraint programming models. This leads to direct CP formulations for both the simple and the multi-cycle variants of the problem, which can serve as baseline models. For details, see Section 3.

Revised and strengthened MIP formulations. We conduct a systematic revision of several MIP formulations from the literature. Our analysis identifies modelling inconsistencies and subtle differences in assumptions. Building on this analysis, we derive strengthened MIP formulations for the basic problem with simple cycles that combine the advantages of existing approaches. For details, see Section 4.

Computational comparison of MIP formulations. We perform extensive computational experiments to evaluate and compare the strength and computational performance of the investigated formulations. The results also clarify the optimal cycle times for several widely used benchmark instances. For details, see Section 5.

A public library for cyclic hoist scheduling. We publish a publicly available library¹ containing benchmark instances and implementations of several CP and MIP formulations. The aim of this library is to support reproducible and comparable research in cyclic hoist scheduling. For details, see Section 6.

2. Hoist scheduling

In this section, we formally define and formulate the hoist scheduling problems considered in this paper. In Section 2.1, we provide a high-level overview to give a global picture, followed by a detailed description in Section 2.2. Table 2 summarizes the notation used throughout the paper.

While defining the problem, we draw on several works from the literature. In particular, papers that provide significant modelling contributions are discussed in Section 2.2. For more comprehensive classifications and surveys, we refer the reader to Manier and Bloch (2003), Dawande et al. (2005), Levner et al. (2010), Feng (2017), Feng et al. (2024) and Laajili (2021) (in French).

2.1. Main overview

Briefly stated, there is an electroplating line, consisting of a set of tanks, in which the parts are processed according to their predefined *processing sequence*. At the *load station*, a part (or a batch of identical parts) is placed into a *carrier* (e.g., a basket), which is then transported

¹ <https://github.com/hmarko89/CyclicHoistScheduling>.

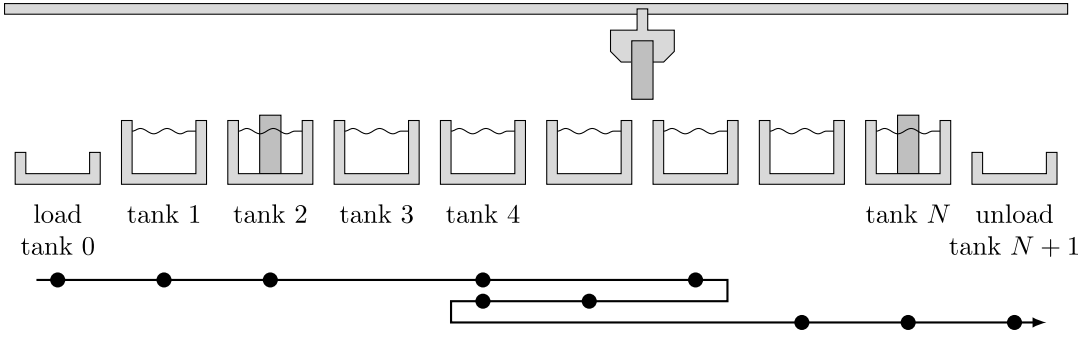


Fig. 1. A bi-directional line with a single hoist. Currently, carriers are soaking in tanks 2 and N , and another one is being transported by the hoist. Tank 3 is not included in this processing sequence, however, the carrier re-enters multi-function tank 4.

Table 2
Notation for hoist scheduling problems.

Parameter	Domain	Description
N	$\mathbb{Z}_{1\leq}$	Number of chemical baths.
n	$\mathbb{Z}_{1\leq}$	Number of soaking operations.
L_i	$\mathbb{R}_{0\leq}$	Minimum duration for soaking operation i .
U_i	$\mathbb{R}_{0\leq}$	Maximum duration for soaking operation i , may be ∞ .
L_0	$\mathbb{R}_{0\leq}$	Minimum duration to load a carrier to the load station.
L_{n+1}	$\mathbb{R}_{0\leq}$	Minimum duration to unload a carrier from the unload station.
s_i	$\{0, 1, \dots, N+1\}$	The tank of operation i .
S_i	$\subseteq \{1, \dots, N\}$	The tanks of soaking operation i (multi-tank case).
$\text{cap}(s_i)$	$\mathbb{Z}_{1\leq}$	Capacity of tank s_i .
d_i	$\mathbb{R}_{0\leq}$	Duration of move i .
$\tilde{e}_{i,j}$	$\mathbb{R}_{0\leq}$	Empty travel time from tank i to tank j .
$e_{i,j}$	$e_{i,j} = \tilde{e}_{s_i, s_j}$	Empty travel time between the tanks of operation i and j .

by a single or multiple *hoists* between the tanks to soak according to the processing sequence, and is finally unloaded at the *unload station*, see Fig. 1. The hoists are identical and move on a single track. The hoist scheduling problem deals with the scheduling of the movements of the hoists. In the cyclic hoist scheduling problem, the goal is to determine a repetitive sequence of hoist movements to process a larger quantity of identical parts. The carriers are successively loaded onto the load station, and unloaded from the unload station.

In the rest of the paper, we consider cyclic problems. For completeness, we mention that Manier and Bloch (2003) distinguished three other problem classes, which are out of the scope of this paper. *Predictive* hoist scheduling problems comprise static problems in which a non-cyclic schedule is sought; for example, scheduling a part through the line to minimize the makespan (e.g., El Amraoui and Elhafsi (2016)). In a *dynamic* hoist scheduling problem, the schedule must be recomputed whenever a new part enters the line (e.g., Feng et al. (2015) and Yan et al. (2017)). In a *reactive* hoist scheduling problem, decisions are made online by assigning tasks to hoists without a complete predefined schedule (e.g., Jégou et al. (2006)).

2.2. Problem definition

We consider a single line with *dissociated* load and unload stations and index the tanks from 0 to $N+1$ following the track. That is, tank 0 is the load station, tanks $1, \dots, N$ are the chemical baths (or *soaking tanks*), and tank $N+1$ is the unload station, see Fig. 1. We consider a single *part type* with a processing sequence consisting of $n+2$ operations, indexed from 0 to $n+1$. The first operation refers to loading the carrier to the load station $s_0 = 0$, and the last operation refers to unloading the carrier from the unload station $s_{n+1} = N+1$. The *soaking operations* $1, \dots, n$ refer to soaking the carrier in their designated tanks $s_1, \dots, s_n$, respectively. Each soaking tank is a *monofunction tank*, that is, used to process only one operation of the processing sequence (i.e., $s_i \neq s_j$ for $1 \leq i < j \leq n$). Only one carrier can soak in a tank at a time. Associated load and unload stations, multiple part types, multi-function tanks, and multi-capacity tanks will be discussed later.

Operations must be processed without preemption. The duration (or *soak time*) of a soaking operation i ($1 \leq i \leq n$) is bounded by minimum and maximum values, L_i and U_i , respectively, which must be respected for quality reasons. The minimum and maximum durations may coincide, i.e., $L_i = U_i$ for each soaking operation i (e.g., Che et al. (2002), Leung and Levner (2006) and Che et al. (2010)); these *fixed processing times* are typical in the case of robot scheduling problems (e.g. Levner et al. (1997) and Yan et al. (2010)). There might also be lower bounds L_0 and L_{n+1} for loading and unloading, respectively, indicating the minimum required time to load/unload a carrier; however, upper bounds may have little practical use ($U_0 = U_{n+1} = \infty$). Loading and unloading are not the responsibility of the hoists. Loading and unloading might be performed simultaneously (which typically holds for dissociated load and unload stations) or not (which typically holds for associated load and unload stations). Operations are subject to no-wait constraints; that is, once a part has been processed in a tank, it must be immediately transported to the next tank.

A hoist can only transport one carrier at a time. In case of multiple hoists, the hoists are identical and move on the same track. The empty travel time between tanks i and j is denoted by $\tilde{e}_{i,j}$. The travel times are symmetric and satisfy the triangle inequality. A *move* is a loaded travel in which a hoist transports a carrier from one tank to the next. A move actually consists of several sub-actions (e.g., lifting the carrier, dripping, transporting the carrier to the next tank, and lowering the carrier), however, the explicit modelling of these sub-actions is only necessary in the multi-hoist case to avoid collisions (e.g., Che et al. (2013)). The duration of move i , i.e., the move associated with operation i ($0 \leq i \leq n$), is denoted by d_i . Note that $\tilde{e}_{s_i, s_{i+1}} \leq d_i$. Moves cannot be interrupted, and their durations are fixed. That is, once a move starts (i.e., the corresponding operation ends), the carrier must be transported to the next tank without any intermediate pause due to technological reasons. Very rarely, move durations are considered as lower bounds rather than exact values (e.g., Liu et al. (2002) and El Amraoui et al. (2008)).

In the cyclic hoist scheduling problem, a repetitive sequence of hoist movements is sought, called a *cycle*. Based on the number of hoists,

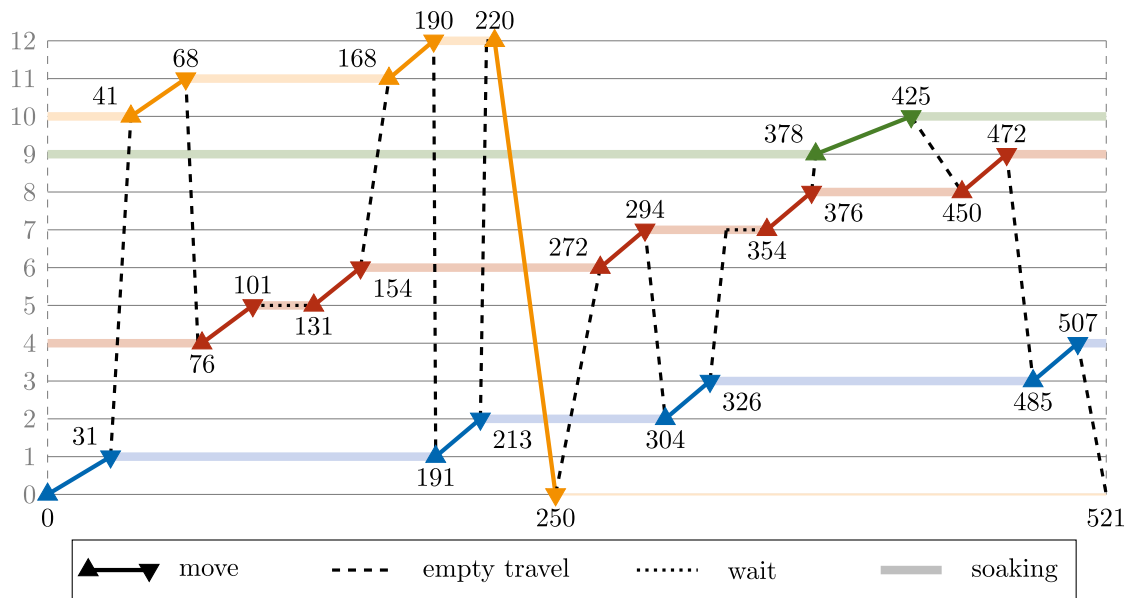


Fig. 2. An optimal single-hoist schedule ($C = 521$) for the instance of Phillips and Unger (1976) with associated load and unload stations. The colours refer to the four carriers used in the simple cycle.

we have a *single-hoist cyclic scheduling problem* (SHCSP) or a *multi-hoist cyclic scheduling problem* (MHCSP). In each cycle, a new carrier enters the line, which leaves the line possibly several cycles later. Multi-cycles are discussed later. The objective is typically to minimize the *cycle time* (or *period*), i.e., the length of the cycle. Other objectives have also been considered (e.g., Xu and Huang (2004) minimized freshwater consumption), sometimes as secondary objectives (e.g., Che et al. (2015) maximized the robustness of the schedule, Feng et al. (2014) and Yan et al. (2016) minimized the travel time of the hoist). For the multi-hoist case, if the number of hoists is not fixed in advance, it may also be treated as an objective to minimize (e.g., Leung and Levner (2006)).

A solution to a cyclic hoist scheduling problem consists of three parts: (i) each move must be assigned to exactly one hoist; (ii) for each hoist, the execution order and timing of the assigned moves must be determined; and (iii) the movements between consecutive moves must be specified. The solution can be illustrated by a *time-way diagram*. In the single-hoist case, the move-hoist assignment is trivial; and it is easy to see that, in addition to the loaded travel, two types of movements are sufficient: the *empty travel* of the hoist to the tank of the next move; and idle *waiting* at a tank. Fig. 2 indicates the time-way diagram of an optimal single-hoist solution for the instance proposed in Phillips and Unger (1976). Note that in each cycle, four carriers are in use, and each carrier leaves the line three cycles after it enters.

In summary, the *basic* single-hoist cyclic scheduling problem consists of a single line, and a simple cycle schedule is sought for a single hoist to process a single part type, where each operation is associated with a single-capacity monofunction tank. Some variants and possible extensions of the basic problem are discussed in the following.

Associated load and unload stations. The load and unload stations may be *associated*, that is, loading and unloading are performed at the same station (i.e., $s_{n+1} = s_0$). However, in the single-hoist case, problems with associated load and unload stations can be converted into dissociated ones without changing the set of feasible solutions or the optimal cycle time. This can be achieved by introducing a fictive unload station with the same travel times as the load station. For details, see Section 3.1.

Multi-function tanks. A soaking tank is a *multi-function tank* (or *reentrant tank*) if it is used to process multiple operations of the same processing

sequence (i.e., $s_i = s_j$ for some $0 < i < j - 1 < n$) (e.g., Liu et al. (2002) and Feng et al. (2018)). The operations associated with the same tank cannot overlap. In some papers, using the same tank for different operations is called *reentrance* or *recirculation*.

Multi-degree problems, multi-part problems. In contrast to a *simple cycle* (or *1-degree cycle*), in an *r-degree cycle* (or *r-cycle*), r carriers enter and leave the line in each cycle (e.g., Che et al. (2011), Li and Fung (2014) and Mao et al. (2018)).

The multi-part case is analogous. Multiple part types must be processed simultaneously on the same line, so that in each cycle, one carrier for each part type enters the line. It is also possible that the different part types do not enter in a 1:1 ratio within a cycle, but rather according to a prescribed production ratio (e.g., Zhao et al. (2013)). In a flow shop environment, the different part types follow the same processing sequence (e.g., El Amraoui et al. (2008, 2012, 2013)); in a job shop environment, each part type has its own processing sequence (e.g., Feng et al. (2018)).

Multi-tank (multi-stage) operations, duplicated tanks. Operations with relatively long minimum durations may create bottlenecks. In such cases, *duplicated tanks* (or *parallel tanks*) are introduced to improve the throughput of the line (e.g., Shapiro and Nuttle (1988)). That is, a *multi-tank (multi-stage) operation* i is associated not with a single tank s_i , but with a set $S_i = \{s_i^1, \dots, s_i^m\}$ of tanks, and each carrier must be soaked in exactly one of them. This setting is similar to the case where the operation is associated not with multiple single-capacity tanks, but with a single *multi-capacity tank*, which allows multiple carriers to soak simultaneously.

Most often, multi-tank operations are handled within a simple cycle (e.g., Zhou and Li (2003)). In each cycle, one carrier enters the line. For a multi-tank operation i , the carrier is assigned to an available tank in S_i , typically the one most recently vacated. The carrier is removed several cycles later, allowing it to soak over multiple cycles. Carriers are removed from the tanks in S_i in the same order in which they were inserted. In some cases, the number of tanks used for a multi-tank operation is not fixed but treated as a decision variable (e.g., Liu et al. (2002), El Amraoui et al. (2012) and Feng et al. (2018)).

2.3. Related literature

In the past decades, many variants of hoist scheduling problems have been investigated, for which several solution approaches have

Table 3
Selected MIP formulations for single-hoist scheduling problems from the literature.

Paper	Degree	Load-unload	Multi-function	Multi-tank
Phillips and Unger (1976)	simple cycle	–	(yes)	–
Liu et al. (2002)	simple cycle	associated	yes	variable
Zhou and Li (2003)	simple cycle	associated	(yes)	fixed
Leung et al. (2004)	simple cycle	–	–	–
Feng et al. (2014)	simple cycle	–	–	–
Che et al. (2015)	simple cycle	–	–	–
El Amraoui and Nait-Sidi-Moh (2012)	simple cycle	associated	yes	variable
Nait-Sidi-Moh and El-Amraoui (2016)	simple cycle	associated	yes	variable
Zhou et al. (2012)	multi-cycle	–	–	–
Li and Fung (2014)	multi-cycle	–	yes	–
El Amraoui et al. (2008)	2-part	–	–	–
El Amraoui et al. (2012)	2-part	associated	yes	variable
El Amraoui et al. (2013)	multi-part	–	–	–
Feng et al. (2018)	multi-part	–	yes	variable

been developed. For an overview, we refer the reader to the recent survey of Feng et al. (2024). In the following, we review some papers that provide formulations for single-hoist problems and are thus relevant to our primary goal: to develop a unified modelling framework for single-hoist cyclic scheduling. Table 3 summarizes the main features of the reviewed MIP formulations.

Phillips and Unger (1976) proposed the first MIP formulation to minimize the cycle time for a single part type in a simple cycle. The authors also extended their model to multi-function tanks. We revise the formulation in Section 4.1, where we show that it is too restrictive; therefore, the resulting solution is not actually optimal in all cases.

Shapiro and Nuttle (1988) developed a branch-and-bound solution procedure for the same problem setting. Their model can be applied to lines with duplicated tanks, where all these tanks must be used in a solution, and can be generalized to determine multi-cycles.

Ng (1995) considered the scenario where the number of duplicated tanks is not fixed. The authors iteratively applied the solution method of Shapiro and Nuttle (1988) to determine the optimal number of duplicated tanks.

Liu et al. (2002) pointed out that the multi-function tank constraints of Phillips and Unger (1976) do not prevent operations using the same tank from overlapping. The authors proposed a corrected formulation for the problem, revised in Section 4.5. They considered a scenario where the move times are not exact, but lower bounds. The authors also extended their formulation to multi-tank stages, where the number of tanks used for a multi-tank operation is a decision variable.

Zhou and Li (2003) extended the model of Phillips and Unger (1976) to multi-tank operations, where the number of tanks used for a multi-tank operation is fixed. The formulation, revised in Section 4.3, uses a correct calculation of the cycle time. The corresponding multi-function tank constraints are not specified in that paper.

Leung et al. (2004) tightened the MIP formulation of Phillips and Unger (1976), and introduced a set of valid inequalities for the convex hull of the feasible solutions, see Section 4.2. However, similarly to the original model, this formulation does not guarantee optimal solutions in certain cases.

El Amraoui and Nait-Sidi-Moh (2012) and Nait-Sidi-Moh and El-Amraoui (2016) proposed MIP formulations for the basic problem as well as extensions to multi-function tanks and multi-tank operations.

Feng et al. (2014) considered minimizing the travel time of the hoist as a secondary objective assuming non-Euclidean travel times.

Che et al. (2015) adjusted the formulation of Liu et al. (2002) to a bi-objective MIP formulation for a robust optimization problem. This model was then used to minimize the cycle time for a given level of robustness, where delays may occur in the execution of the moves or in the empty travels.

Wallace and Yorke-Smith (2020) proposed a non-linearized CP formulation capable of handling multiple tank capacities.

Liu et al. (2025) emphasized the substantial potential of modern constraint programming solvers for hoist scheduling problems. The

authors transformed the classical single-hoist MIP formulation (cf. Zhou and Li (2003)) into a CP formulation, and solved the widely-used benchmark instances with OR-Tools CP-SAT. However, based on the reported cycle times (see Table 1), there appear to be discrepancies: either the implementation is incorrect, or the benchmark dataset differs from those available in the literature. For further discussion, we refer to Section 5.1.1.

Zhou et al. (2012) developed the first MIP formulation for processing a single part type in multi-cycles. The model of Li and Fung (2014) is applicable in the presence of multi-function tanks. These two papers follow different modelling approaches, which we will discuss later in Section 3.3.

El Amraoui et al. (2008) proposed an MIP formulation for two part types with the same processing sequence. The model of El Amraoui et al. (2013) is applicable for an arbitrary number of part types that follow the same processing sequence. El Amraoui et al. (2012) provided extensions for the 2-part formulations, including multi-function tanks and multi-tank operations.

Zhao et al. (2013) considered a multi-part hoist scheduling problem in which the number of carriers entering the line per cycle for each part type is determined according to prescribed production ratios.

Feng et al. (2018) developed an MIP formulation for multiple part types with different processing sequences, which may contain multi-capacity multi-function tanks.

Some works are closely related to our primary goal, as they introduce systematic modelling approaches. However, our approach for the single-hoist case covers a broader range of extensions and additionally provides reproducible formulations and benchmark instances to support systematic comparison. El Amraoui and Nait-Sidi-Moh (2012) and Nait-Sidi-Moh and El-Amraoui (2016) used P-temporal Petri Nets to formally model and analyse single-hoist cyclic schedules for the basic problem, as well as for extensions involving multi-function tanks and multi-tank operations. Recently, Feng et al. (2024) provided a comprehensive review of robotic cell scheduling. In addition, the authors drew on several of these modelling approaches to present an MIP formulation for the basic problem and for extensions such as multi-function tanks, multi-tank operations, multi-cycle settings, and multiple hoists.

3. Modelling single-hoist cyclic scheduling problems

In this section, we address the modelling of single-hoist cyclic scheduling problems. In Section 3.1, we introduce the necessary definitions and notation. Sections 3.2 and 3.3 present the formulations for simple cycles and multi-cycles, respectively. Before that, we describe the considered problem settings using two complementary classification schemes.

First, we characterize the problem settings according to the classification of Manier and Bloch (2003) (with the corresponding notation in parentheses). We consider a single line ($\beta_1 = \beta_2 = \beta_3 = \emptyset$), a single hoist ($\beta_{41} = \emptyset$), and an arbitrary number of soaking tanks ($\beta_{42} = N$) with unit

Table 4
Notation for the single-hoist formulations.

Variables	Domain	Description
t_i	$\mathbb{R}_{0 \leq}$	Start time of move i ($0 \leq i \leq n$).
C	$\mathbb{R}_{0 \leq}$	Cycle time.
$y_{i,j}$	$\{0, 1\}$	Indicates whether move j starts after move i ($0 \leq i, j \leq n, i \neq j$).
z_i	$\{0, 1\}$	Indicates whether move i is the latest move ($0 \leq i \leq n$).
$t_{\max}$	$\mathbb{R}_{0 \leq}$	Maximum start time among all moves.

capacity ($\beta_{43} = \emptyset$). We also consider an extension in which bottleneck operations are associated with multi-capacity tanks ($\beta_{43} = ct$). The number of available carriers can be either unlimited or limited ($\beta_5 \in \{\emptyset, nc\}$), and a storage is available for them; thus, empty carriers do not remain on the line ($\beta_8 = \emptyset$). Load and unload stations can be either associated or dissociated ($\beta_9 \in \{\emptyset, ass, diss\}$), when relevant. An infinite number of identical parts are processed on the line ($\delta_1 = \emptyset, \delta_2 = \emptyset$), with an arbitrary number of soaking operations ($\delta_3 = n$), which may use the same tanks multiple times ($\delta_5 \in \{\emptyset, recr\}$). After being unloaded, the carriers do not require cleaning ($\delta_4 = \emptyset$). We also consider the multi-part case ($\delta_2 = nps$). The objective is to find a cyclic schedule ($\alpha = CHSP$) with a minimum cycle time ($\gamma = Tmin$), however, our modelling approach is mostly independent of the objective function.

Next, we describe the problem settings using the more recent classification of Feng et al. (2024). **Tank settings:** in addition to basic systems, we consider extended systems with reentrant tanks, parallel tanks, or both. **Number of hoists:** a single hoist is used to transport the carriers. **Part processing time:** each operation is associated with a processing time window, where the minimum and maximum durations may coincide (i.e., fixed processing times). **Part processing routine:** in addition to the single-part case (single-part flow shop), we consider the multi-part extension, where different part types may follow different processing sequences (multiple part-type flow shop or job shop). **Scheduling mode:** we consider both one-degree cyclic scheduling (i.e., simple cycles) and multi-degree cyclic scheduling. In the latter case, either a single part type is processed in a multi-cycle, or multiple part types are processed within a single cycle. **Optimization criteria:** our modelling approach is mostly independent of the objective function; however, in all considered formulations, the cycle time is minimized.

3.1. Preliminaries

Table 4 summarizes the notation that is used throughout Sections 3 and 4. The continuous *start time variable* t_i denotes the start time of the move i ($0 \leq i \leq n$), and the *cycle time variable* C denotes the cycle time. The binary *ordering variable* $y_{i,j}$ indicates whether move j starts after move i ($0 \leq i < j \leq n$), i.e., $t_i < t_j$. Some formulations may interpret variables $y_{i,j}$ with $i > j$ as well. Clearly, $y_{j,i} = 1 - y_{i,j}$ holds for all $i \neq j$. We call a formulation a *base formulation* if it uses only the variables C , t_i , and $y_{i,j}$. An *extended formulation* uses additional variables as well. The binary variable z_i indicates whether the move i is the latest move ($0 \leq i \leq n$), and the continuous variable $t_{\max}$ denotes the maximum start time among all moves (i.e., the start time of the latest one).

Without loss of generality, we assume that the cycle starts with the immediate execution of move 0, that is, $t_0 = 0$. This implies that $y_{0,j} = 1$ ($1 \leq j \leq n$) and $z_0 = 0$. Moreover, this means that no move crosses the cycle boundary, that is, $t_i + d_i \leq C$ holds for each move i .

To simplify the notation, let $e_{i,j} = \bar{e}_{s_i, s_j}$ denote the empty travel time from the tank of operation i to the tank of operation j . Recall that the travel times satisfy the triangle inequality, and that $e_{i,i+1} \leq d_i$ holds ($0 \leq i \leq n$); both properties are used in the formulations. In some constraints, we use an appropriately large constant M , which must be an upper bound on the cycle time. For this purpose, we apply the upper bound (1) defined in the next subsection.

As defined in Section 2.2, we consider dissociative load and unload stations (i.e., $s_0 = 0$ and $s_{n+1} = N + 1$). This is not a restriction: in

the single-hoist case, any problem instance with associative load and unload stations (i.e., $s_0 = s_{n+1} = 0$) can be converted to an equivalent instance with dissociative stations without changing the set of feasible solutions or the optimal cycle time. Specifically, a fictive unload station, $N + 1$, is introduced for the unloading operation such that $\bar{e}_{0, N+1} = 0$ and $\bar{e}_{0,j} = \bar{e}_{N+1,j} = 0$ ($0 \leq j \leq N$), i.e., the travel times from/to the load station are identical to those from/to the unload station.

3.1.1. Bounds on the cycle time

We present some lower and upper bounds on the cycle time for the case of minimizing the cycle time. Note, however, that due to some loading/unloading configurations (see Section 3.2.4) or due to multi-tank operations (see Section 3.2.6), these bounds may need to be adjusted.

One can construct a primitive feasible solution for the problem in the following way. The hoist executes the moves in the prescribed processing order. When the carrier is placed in a tank, the hoist waits there until the corresponding operation is finished with its minimum processing duration. After the execution of the last move, the hoist travels back to the load station. This schedule provides an upper bound on the optimal cycle time, see (1), which will be used as the big-M constant or variable upper bounds in the formulations.

$$UB = d_0 + \sum_{i=1}^n (L_i + d_i) + e_{n+1,0} \quad (1)$$

The hoist trajectory includes all moves and at least one empty travel. Thus, the sum of move durations plus the minimum travel time to the load station is a lower bound on the cycle time. For each soaking operation i , the following are disjoint events: execution of move $i - 1$, processing of operation i , execution of move i , travelling (indirectly) from s_{i+1} to s_{i-1} . Thus, the total minimum duration of these events provides another lower bound on the cycle time. These two observations can be combined in a single lower bound, see (2).

$$LB = \max \left\{ \min_{1 \leq i \leq n} \{d_{i-1} + L_i + d_i + e_{i+1, i-1}\}, \min_{0 \leq i \leq n} \{e_{i+1, 0}\} + \sum_{i=0}^n d_i \right\} \quad (2)$$

3.1.2. Time-way diagram

The solution, i.e., the determined hoist movements, can be illustrated in a time-way diagram. It is easy to see that in the single-hoist case, no other movements are needed in addition to the following ones: (i) the hoist executes a move; (ii) the hoist travels empty from the tank where the previous move is finished to the tank where the next move starts; (iii) the hoist waits either at the tank where the previous move is finished or at the tank where the next move starts. Based on these observations, the start times of the moves and the cycle time are sufficient to describe a solution for a single-hoist cyclic scheduling problem.

For given move start times $t_0, \dots, t_n$ and cycle time C , the following procedure constructs the time-way diagram.

Step 1 (Operation start times). Let $t_0^+ = t_n + d_n$ be the start time of unloading. Let $t_i^+ = t_{i-1} + d_{i-1} \pmod{C}$ be the start time of operation i ($1 \leq i \leq n$), that is, if $C \leq t_{i-1} + d_{i-1}$, then $t_i^+ = t_{i-1} + d_{i-1} - C$.

Step 2 (Execution order). Let π denote the execution order of moves, that is, $0 = t_{\pi_0} < t_{\pi_1} < \dots < t_{\pi_n} < C$.

Step 3 (Trajectory). Following the execution order π , we construct the parts of the hoist's trajectory. For the incumbent move π_i :

- (1) Time interval $[t_{\pi_i}, t_{\pi_i} + d_{\pi_i}]$: the hoist executes move π_i .
- (2) Time interval $[t_{\pi_i} + d_{\pi_i}, t_{\pi_i} + d_{\pi_i} + e_{\pi_i+1, \pi_{i+1}}]$ (may be empty): the hoist travels empty from tank s_{π_i+1} to tank $s_{\pi_{i+1}}$, where $s_{\pi_{i+1}}$ denotes the unload station. Note that the time interval is empty if $s_{\pi_i+1} = s_{\pi_{i+1}}$.
- (3) Time interval $[t_{\pi_i} + d_{\pi_i} + e_{\pi_i+1, \pi_{i+1}}, t_{\pi_{i+1}}^+]$ (may be empty): the hoist waits at tank $s_{\pi_{i+1}}$, where $t_{\pi_{i+1}}^+ = C$.

3.2. Modelling simple cycles

A valid formulation for a single-hoist cyclic hoist scheduling problem with simple cycles must enforce four basic constraints:

- There must be enough time for empty travel between the consecutive moves, see Section 3.2.1.
- The cycle time must be long enough for the hoist to return to the load station after the latest move, see Section 3.2.2.
- Minimum and maximum soak times must be respected, see Section 3.2.3.
- Loading and unloading times must be respected, see Section 3.2.4.

We also consider the following extensions:

- In case of multi-function tanks, operations using the same tank cannot overlap, see Section 3.2.5.
- Operations associated with multiple single-capacity tanks or a single multi-capacity tank, may be soaked for several cycles, see Section 3.2.6.
- The number of carriers used in a cycle may be limited, see Section 3.2.7.

3.2.1. Travel time constraints

There must be enough time between consecutive moves to allow the hoist to execute the former one and then travel empty to the next tank. That is, if move j is the immediate successor of move i , then $t_i + d_i + e_{i+1, j} \leq t_j$ must hold. This can be enforced, for example, by the following indicator constraints (3)–(4).

$$y_{i,j} = 1 \rightarrow t_j \geq t_i + d_i + e_{i+1, j} \quad 0 \leq i < j \leq n \quad (3)$$

$$y_{i,j} = 0 \rightarrow t_i \geq t_j + d_j + e_{j+1, i} \quad 0 \leq i < j \leq n \quad (4)$$

These constraints imply the following lower bound on the start time variables: $t_j \geq d_0 + e_{1, j}$ ($1 \leq j \leq n$), since $t_0 = 0$ and $y_{0, j} = 1$. Moreover, the actual interpretation of the y -variables is the following: $y_{i, j} = 1$ if $t_i + d_i \leq t_j$, and $y_{i, j} = 0$ if $t_j + d_j \leq t_i$.

In an MIP formulation, for example, the linearized constraint (5) can be used:

$$t_j \geq t_i + d_i + e_{i+1, j} - M(1 - y_{i, j}) \quad 0 \leq i, j \leq n : i \neq j \quad (5)$$

where M is an appropriate big constant (an upper bound on the cycle time) as described in Section 3.1.

3.2.2. Cycle time constraints

The cycle time must be long enough for the hoist to return to the load station after the latest move. Constraint (6) ensures that this is fulfilled for every move.

$$C \geq t_i + d_i + e_{i+1, 0} \quad 0 \leq i \leq n \quad (6)$$

Note that constraint (6) implies that each move starts and ends within the same cycle, that is, $t_i + d_i \leq C$ ($0 \leq i \leq n$).

3.2.3. Soaking constraints

Minimum and maximum soak times must be respected. For a soaking operation i ($1 \leq i \leq n$), there are two possible scenarios:

- (i) $t_{i-1} + d_{i-1} \leq t_i$: operation i starts and ends within the same cycle, see Fig. 3(a). The soak time is $t_i - (t_{i-1} + d_{i-1})$.
- (ii) $t_i < t_{i-1} + d_{i-1}$: operation i is in process at the beginning of the cycle, see Fig. 3(b). The soak time is $(C + t_i) - (t_{i-1} + d_{i-1})$.

Variable $y_{i-1, i}$ is suitable to distinguish these two cases for operation i , see constraints (7)–(8).

$$y_{i-1, i} = 1 \rightarrow L_i \leq t_i - (t_{i-1} + d_{i-1}) \leq U_i \quad 1 \leq i \leq n \quad (7)$$

$$y_{i-1, i} = 0 \rightarrow L_i \leq (C + t_i) - (t_{i-1} + d_{i-1}) \leq U_i \quad 1 \leq i \leq n \quad (8)$$

Chen et al. (1998) expressed these constraints with non-linear inequalities: $L_i \leq (y_{i-1, i} C + t_i) - (t_{i-1} + d_{i-1}) \leq U_i$ ($1 \leq i \leq n$). In an MIP formulation, constraints (7) and (8) can be linearized, for example, using big-M formulations, see constraints (9) and (10), respectively.

$$L_i - M(1 - y_{i-1, i}) \leq t_i - (t_{i-1} + d_{i-1}) \leq U_i \quad 1 \leq i \leq n \quad (9)$$

$$L_i \leq (C + t_i) - (t_{i-1} + d_{i-1}) \leq U_i + M y_{i-1, i} \quad 1 \leq i \leq n \quad (10)$$

As Leung et al. (2004) pointed out, no big-M term is needed for the upper bound inequality of constraint (9) and for the lower bound inequality of constraint (10), since the corresponding inequalities hold in both of the two scenarios. Our computational experiments in Section 5 will also demonstrate the strength of these improvements.

3.2.4. Loading and unloading constraints

Loading and unloading times must be respected. We consider the following two load-unload configurations:

- (i) Dissociated load and unload stations: the loading and unloading operations can be performed simultaneously.
- (ii) Associated load and unload stations: the loading and unloading operations cannot be performed simultaneously. Each carrier must be unloaded from the load-unload station before the next one can be loaded.

In case of dissociated stations, the cycle time must be long enough to perform either operation:

$$\max\{L_0, L_{n+1}\} \leq C \quad (11)$$

Although such upper bounds may be of limited practical relevance, they can also be handled easily: $C \leq \min\{U_0, U_{n+1}\}$.

In case of associated stations, there must be enough time for unloading the carrier and loading the next one:

$$(t_n + d_n) + L_{n+1} + L_0 \leq C \quad (12)$$

One can assume that $L_{n+1} = 0$, since the unloading time can be merged into the loading time. Note that in the presence of constraint (12), the upper bound (1) must be adjusted by replacing the term $e_{n+1, 0}$ with $\max\{e_{n+1, 0}, L_0 + L_{n+1}\}$. If upper bound U_0 is also given, then constraint $C \leq (t_n + d_n) + U_0$ can be used. However, if $e_{n+1, 0} < U_0$, then the primitive schedule is not feasible, and thus the upper bound (1) is not valid.

3.2.5. Multi-function tanks

The operations processed in the same multi-function tank cannot overlap. Assume that operations i and j ($0 < i < j - 1 < n$) require the same tank (i.e., $s_i = s_j$). The former constraints ensure that an appropriate amount of time elapses between moves. The remaining task is to ensure that the start and finish times of the operations are in the correct cyclic order. There are four possible scenarios:

- (i) $t_{i-1} + d_{i-1} < t_i < t_{j-1} + d_{j-1} < t_j$: both operations start and end within the cycle, operation i starts earlier;

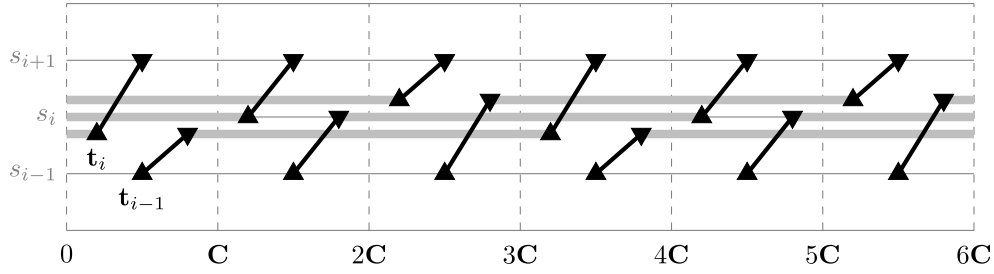


Fig. 5. The process of operation i follows a 3-period pattern. The carrier inserted into the tank in a given cycle is removed three cycles later, after being soaked for a duration between $2C$ and $3C$.

Lower and upper bounds on the cycle time. Since the soak time of operation i is between $(m-1)C$ and mC , $(m-1)C \leq U_i$ and $L_i \leq mC$ must hold, otherwise the problem is infeasible. This observation gives the following lower and upper bounds on the cycle time:

$$\frac{L_i}{m} \leq C \leq \frac{U_i}{m-1} \quad (19)$$

The bounds (1)–(2) on the cycle time must be adjusted accordingly.

Duplicated tanks. If the bottleneck operation i is associated with multiple single-capacity tanks (duplicated tanks) instead of a single multi-capacity tank, the situation can be handled similarly. The group $S_i = \{s_i^1, s_i^2, \dots, s_i^m\}$ of tanks where operation i can be processed can be merged into a single m -capacity tank, say s_i' . Each tank can belong to at most one group.

The advantage of this approach is that all carriers have identical soak times, ensuring consistent production quality. However, move and travel times must be adjusted, which may lead to suboptimal solutions. For instance, if $\tilde{e}_{s_i', s_j} = \max_{1 \leq k \leq m} \{\tilde{e}_{s_i^k, s_j}\}$ is used, i.e., the largest travel time among the tanks in the group, additional waiting times may be required when the hoist travels to or from tanks that are closer than the furthest one (Shapiro and Nuttle, 1988).

An alternative way of handling duplicated tanks will be discussed in Section 3.3.

Determining the optimal number of duplicated tanks. The number m of duplicated tanks used for operation i (i.e., the utilized portion of the capacity $\text{cap}(s_i)$) may also be treated as a decision variable. Let the binary variable $u_{i,m}$ indicate whether exactly m units of the tank capacity are used ($1 \leq m \leq \text{cap}(s_i)$). Of course, exactly one value must be selected:

$$\sum_{m=1}^{\text{cap}(s_i)} u_{i,m} = 1 \quad (20)$$

Now, the soak time of the operation also depends on the chosen capacity level:

$$y_{i-1,i} = 1 \text{ and } u_{i,m} = 1 \rightarrow L_i \leq ((m-1)C + t_i) - (t_{i-1} + d_{i-1}) \leq U_i \quad 1 \leq m \leq \text{cap}(s_i) \quad (21)$$

$$y_{i-1,i} = 0 \text{ and } u_{i,m} = 1 \rightarrow L_i \leq (mC + t_i) - (t_{i-1} + d_{i-1}) \leq U_i \quad 1 \leq m \leq \text{cap}(s_i) \quad (22)$$

For an MIP formulation, the constraints can be linearized as:

$$L_i - M(2 - y_{i-1,i} - u_{i,m}) \leq ((m-1)C + t_i) - (t_{i-1} + d_{i-1}) \leq U_i + M(1 - u_{i,m}) \quad 1 \leq m \leq \text{cap}(s_i) \quad (23)$$

$$L_i - M(1 - u_{i,m}) \leq (mC + t_i) - (t_{i-1} + d_{i-1}) \leq U_i + M(y_{i-1,i} + 1 - u_{i,m}) \quad 1 \leq m \leq \text{cap}(s_i) \quad (24)$$

Similar to the fixed-capacity scenario, move and travel times may require adjustment based on the number of tanks used (Liu et al., 2002).

3.2.7. Limited number of carriers

The number of carriers available for production may be limited (e.g., Wallace and Yorke-Smith (2020)). Constraint (25) requires that at most k carriers can be used in each cycle, i.e., exactly one carrier enters the line, and at most $k-1$ carriers soak at the beginning of the cycle.

$$1 + \sum_{i=1}^n (1 - y_{i-1,i}) \leq k \quad (25)$$

Note that in case of multi-tank operations, constraint (25) must be adjusted according to the tank capacities. That is, for operation i , the term $(1 - y_{i-1,i})$ must be replaced with $(m - y_{i-1,i})$ or $(\sum_{m=1}^{\text{cap}(s_i)} m u_{i,m} - y_{i-1,i})$, depending on whether the number of tanks/units, m , used for the operation is fixed or a decision variable.

3.2.8. Valid inequalities

As the y -variables define a linear order, each formulation containing these variables may be strengthened by valid inequalities for the linear ordering polytope (see e.g., Grötschel et al. (1985)).

3.2.9. A natural CP formulation for simple cycles

Our modelling considerations for simple cycles naturally lead to a direct CP formulation. The objective, for example, is to minimize the cycle time C , subject to constraints (3)–(4), (6), (7)–(8), where $C \in \{0, \dots, M\}$, $t_0 = 0$, $t_i \in \{0, \dots, M\}$ ($1 \leq i \leq n$), $y_{0,j} = 1$ ($1 \leq j \leq n$), $y_{i,j} \in \{0, 1\}$ ($1 \leq i < j \leq n$). Depending on the load-unload configuration, constraint (11) or (12) can be used. In case of multi-function tanks, constraint (13); in case of multi-tank operations, constraints (20) and (21)–(22) are added to the model.

3.3. Modelling multi-degree cycles

In case of an r -cycle, r carriers enter (and leave) the line, thus each operation i has r copies. The modelling observations in Section 3.2 for simple cycles can be easily adapted for multi-degree problems. In the following, we discuss only the modelling of the basic feasibility constraints and the extension to multi-function tanks.

Let operation (p, i) refer to the p th copy of operation i ($1 \leq p \leq r$, $0 \leq i \leq n+1$). That is, operation (p, i) refers to a carrier that enters the p th position in some cycle. For moves (p, i) and (q, j) , we use the lexicographic order: $(p, i) \leq (q, j)$ if and only if $p < q$, or $p = q$ and $i \leq j$. Let the variable $t_{(p,i)}$ denote the start time of move (p, i) , and let the binary variable $y_{(p,i),(q,j)}$ indicate whether $t_{(p,i)} < t_{(q,j)}$ ($(1, 0) \leq (p, i) < (q, j) \leq (r, n)$). Without loss of generality, we assume that the cycle starts with the execution of move $(1, 0)$, that is, $t_{(1,0)} = 0$. Consequently, $y_{(1,0),(q,j)} = 1$ holds for each $(1, 0) < (q, j)$. Note that $s_{(p,i)} = s_i$, $d_{(p,i)} = d_i$, $e_{(p,i),(q,j)} = e_{i,j}$, $L_{(p,i)} = L_i$ and $U_{(p,i)} = U_i$, however, we use the general notation, because the formulation in the presented form will be easily adaptable to some extensions, including multiple part types, see Section 3.3.7.

3.3.1. Symmetry breaking constraints

Without loss of generality, we can assume that the carriers enter the line in the order $1, \dots, r$:

$$y_{(p,0),(p+1,0)} = 1 \leq p < r \quad (26)$$

However, we do not use this simplification in the following expressions, because the formulation in the presented form will be easily adaptable to the multi-part case, see Section 3.3.7.

3.3.2. Travel time constraints

The travel time constraints are the following (cf. constraints (3)–(4)):

$$y_{(p,i),(q,j)} = 1 \rightarrow t_{(q,j)} \geq t_{(p,i)} + d_{(p,i)} + e_{(q,j+1),(p,i)} \quad (1, 0) \leq (p, i) < (q, j) \leq (r, n) \quad (27)$$

$$y_{(p,i),(q,j)} = 0 \rightarrow t_{(p,i)} \geq t_{(q,j)} + d_{(q,j)} + e_{(q,j+1),(p,i)} \quad (1, 0) \leq (p, i) < (q, j) \leq (r, n) \quad (28)$$

3.3.3. Cycle time constraints

The cycle time must be long enough for the hoist to return to the load station (cf. constraint (6)):

$$C \geq t_{(p,i)} + d_{(p,i)} + e_{(p,i+1),(p,0)} \quad (1, 0) \leq (p, i) \leq (r, n) \quad (29)$$

Note that $s_{(p,0)} = 0$ for all $1 \leq p \leq r$, that is, the loading station is the same for all copies. This also holds for all part types, in the multi-part case, see Section 3.3.7.

3.3.4. Soaking constraints

The soaking constraints are the following (cf. constraints (7)–(8)):

$$y_{(p,i-1),(p,i)} = 1 \rightarrow L_{(p,i)} \leq t_{(p,i)} - (t_{(p,i-1)} + d_{(p,i-1)}) \leq U_{(p,i)} \quad (1, 1) \leq (p, i) \leq (r, n) \quad (30)$$

$$y_{(p,i-1),(p,i)} = 0 \rightarrow L_{(p,i)} \leq (C + t_{(p,i)} - (t_{(p,i-1)} + d_{(p,i-1)})) \leq U_{(p,i)} \quad (1, 1) \leq (p, i) \leq (r, n) \quad (31)$$

However, we must ensure that the different copies of an operation do not overlap. The situation is very similar to the multi-function tank case discussed in Section 3.2.5. Therefore, the following pairwise constraints prohibit overlaps (cf. constraint (13)):

$$y_{(p,i-1),(p,i)} + y_{(p,i),(q,i-1)} + y_{(q,i-1),(q,i)} + (1 - y_{(p,i-1),(q,i)}) = 3 \quad (1, 1) \leq (p, i) < (q, i) \leq (r, n) \quad (32)$$

This implies $y_{(p,i-1),(p,i)} + y_{(q,i-1),(q,i)} \geq 1$ for each pair (p, q) of copies. Consequently, $\sum_{p=1}^r y_{(p,i-1),(p,i)} \geq r - 1$ is a valid inequality.

Note that the no-wait constraints ensure that the entry order is preserved throughout the line, up to a cyclic shift; that is, $(1, 2, \dots, r)$, if the symmetry-breaking constraints provided in Section 3.3.1 are applied. Consequently, a property that does not hold in the multi-function tank case, where operations may correspond to different processing stages, becomes valid here, as all considered operations belong to the same stage. Specifically, feasible sequences for operation i correspond exactly to cyclic shifts of $((1, i), (2, i), \dots, (r, i))$. Therefore, constraint (32) can be replaced by the following one (cf. constraint (14)):

$$\sum_{p=1}^r (y_{(p,i-1),(p,i)} + y_{(p,i),(p+1,i-1)}) = 2r - 1 \quad 1 \leq i \leq n \quad (33)$$

where the substitution $y_{(r,i),(r+1,i-1)} = 1 - y_{(1,i-1),(r,i)}$ is applied to simplify the notation. Now, this constraint implies $\sum_{p=1}^r y_{(p,i-1),(p,i)} \geq r - 1$. Despite this, these constraints are usually used together (e.g., Zhou et al. (2012) and Feng et al. (2024)).

3.3.5. Loading and unloading constraints

We consider the same load–unload configurations as in Section 3.2.4. Under the symmetry-breaking constraints from Section 3.3.1, these constraints admit a simplified form, which we do not detail here.

If the load and unload stations are dissociated, constraints (34)–(37) ensure that there is sufficient time between two first moves, and constraints (38)–(41) enforce the same condition for last moves.

$$y_{(p,0),(q,0)} = 1 \rightarrow t_{(p,0)} + L_{(q,0)} \leq t_{(q,0)} \quad 1 \leq p < q \leq r \quad (34)$$

$$y_{(p,0),(q,0)} = 1 \rightarrow t_{(q,0)} + L_{(p,0)} \leq C + t_{(p,0)} \quad 1 \leq p < q \leq r \quad (35)$$

$$y_{(p,0),(q,0)} = 0 \rightarrow t_{(q,0)} + L_{(p,0)} \leq t_{(p,0)} \quad 1 \leq p < q \leq r \quad (36)$$

$$y_{(p,0),(q,0)} = 0 \rightarrow t_{(p,0)} + L_{(q,0)} \leq C + t_{(q,0)} \quad 1 \leq p < q \leq r \quad (37)$$

$$y_{(p,n),(q,n)} = 1 \rightarrow t_{(p,n)} + d_{(p,n)} + L_{(p,n+1)} \leq t_{(q,n)} + d_{(q,n)} \quad 1 \leq p < q \leq r \quad (38)$$

$$y_{(p,n),(q,n)} = 1 \rightarrow t_{(q,n)} + d_{(q,n)} + L_{(q,n+1)} \leq C + t_{(p,n)} + d_{(p,n)} \quad 1 \leq p < q \leq r \quad (39)$$

$$y_{(p,n),(q,n)} = 0 \rightarrow t_{(q,n)} + d_{(q,n)} + L_{(q,n+1)} \leq t_{(p,n)} + d_{(p,n)} \quad 1 \leq p < q \leq r \quad (40)$$

$$y_{(p,n),(q,n)} = 0 \rightarrow t_{(p,n)} + d_{(p,n)} + L_{(p,n+1)} \leq C + t_{(q,n)} + d_{(q,n)} \quad 1 \leq p < q \leq r \quad (41)$$

Note that every second constraint accounts for cases where loading/unloading operations cross the cycle boundary.

If the load and unload stations are associated, constraints (42)–(45) enforce a minimum gap between dropping off a carrier and picking up another one at the load–unload station.

$$y_{(p,n),(q,0)} = 1 \rightarrow t_{(p,n)} + d_{(p,n)} + L_{(p,n+1)} + L_{(q,0)} \leq t_{(q,0)} \quad 1 \leq p < q \leq r \quad (42)$$

$$y_{(p,n),(q,0)} = 0 \rightarrow t_{(p,n)} + d_{(p,n)} + L_{(p,n+1)} + L_{(q,0)} \leq C + t_{(q,0)} \quad 1 \leq p < q \leq r \quad (43)$$

$$y_{(p,0),(q,n)} = 1 \rightarrow t_{(q,n)} + d_{(q,n)} + L_{(q,n+1)} + L_{(p,0)} \leq C + t_{(p,0)} \quad 1 < p \leq q \leq r \quad (44)$$

$$y_{(p,0),(q,n)} = 0 \rightarrow t_{(q,n)} + d_{(q,n)} + L_{(q,n+1)} + L_{(p,0)} \leq t_{(p,0)} \quad 1 < p \leq q \leq r \quad (45)$$

3.3.6. Multi-function tanks

Assume that operations i and j ($0 < i < j - 1 < n$) are processed in the same tank (i.e., $s_i = s_j$). Then, these operations of the same copy cannot overlap (cf. constraints (13)):

$$y_{(p,i-1),(p,i)} + y_{(p,i),(p,j-1)} + y_{(p,j-1),(p,j)} + (1 - y_{(p,i-1),(p,j)}) = 3 \quad 1 \leq p \leq r \quad (46)$$

These operations of different copies also cannot overlap:

$$y_{(p,i-1),(p,i)} + y_{(p,i),(q,j-1)} + y_{(q,j-1),(q,j)} + (1 - y_{(p,i-1),(q,j)}) = 3 \quad 1 \leq p < q \leq r \quad (47)$$

$$y_{(p,j-1),(p,j)} + y_{(p,j),(q,i-1)} + y_{(q,i-1),(q,i)} + (1 - y_{(p,j-1),(q,i)}) = 3 \quad 1 \leq p < q \leq r \quad (48)$$

3.3.7. Adaptations to simple cycle extensions

Some extensions of the simple cycle problem can be addressed using multi-cycles.

Multi-tank operations. Consider a multi-tank operation i that can be performed in any tank from the group $S_i = \{s_i^1, \dots, s_i^m\}$. Rather than merging S_i into a single m -capacity tank (cf. Section 3.2.6), the operation can be processed using an m -cycle, assigning the k th carrier to the k th tank in S_i . This preserves the actual move and travel times ($d_{(p,i)}$ and $e_{(p,i),(q,j)}$). However, without additional restrictions, the soak times of the different copies may vary, which may be undesirable. If not all tanks must be used, an enumeration procedure may be needed

to select the optimal number of duplicated tanks. A major disadvantage of this approach is that handling multiple multi-tank operations can be inefficient, as the degree of the problem must be a common multiple of the group sizes.

Multiple part types. This model can be easily adapted to multiple part types, where a set of r parts is processed simultaneously. In each cycle, one carrier enters for each part type. Here, (p, i) refers to the i th operation of part type p ($1 \leq p \leq r$, $0 \leq i \leq n_r$), where n_r denotes the number of soaking operations in the processing sequence of part type r . The symmetry-breaking constraints in Section 3.3.1 no longer apply.

3.3.8. Natural CP formulations for multi-degree problems

Our modelling observations for multi-cycles naturally lead to a direct CP formulation for the single-part problem with multi-cycles. The formulation can be easily adapted to the multi-part problem with a single cycle according to Section 3.3.7.

The objective is, for example, to minimize the cycle time C subject to constraints (26), (27)–(28), (29), and (30)–(32), where $C \in \{0, \dots, M\}$, $t_{(1,0)} = 0$, $t_{(p,i)} \in \{0, \dots, M\}$ ($(1,0) < (p,i) \leq (r,n)$), $y_{(1,0),(q,j)} = 1$ ($(1,0) < (q,j) \leq (r,n)$), $y_{(p,i),(q,j)} \in \{0, 1\}$ ($(1,0) < (p,i) \leq (q,j) \leq (r,n)$). Depending on the load-unload configuration, constraints (34)–(41) or (42)–(45) can be used. In case of multi-function tanks, constraints (46)–(48) are added to the model. The constant M can be chosen, for example, as r times the upper bound (1) of the simple cycle.

3.3.9. Other modelling approaches

The MIP formulation of Li and Fung (2014) followed a different modelling approach. In their formulation, move (p, i) refers to the k th copy of operation i in chronological order within the cycle. That is, $t_{(1,i)} < t_{(2,i)} < \dots < t_{(r,i)}$ holds for each operation i ($0 \leq i \leq n$). This ordering allowed certain constraints to be expressed more easily, but it was more cumbersome to generalize to multiple part types. However, the proposed multi-function tank constraints are relatively complex.

4. Revised and improved MIP formulations for simple cycles

In this section, we revise several MIP formulations from the literature for the basic problem. That is, the problem setting considered consists of a single line with a single part type and a single hoist, an arbitrary number of single-capacity tanks, and an arbitrary number of operations, possibly with different load-unload configurations. A simple cycle is sought with a minimal cycle time.

In Sections 4.1–4.3, we present the evolution of the extended formulations for the problem, including the models of Phillips and Unger (1976), Leung et al. (2004) and Zhou and Li (2003). Combining their advantages, we present an improved formulation in Section 4.4. In Section 4.5, we revise the base formulation of Liu et al. (2002), and we propose a strengthened version in Section 4.6. The comparison of the investigated formulations is presented in Section 5.

4.1. Formulation of Phillips and Unger (1976)

Phillips and Unger (1976) aimed to minimize the cycle time, for which they used three sets of constraints. Although the authors did not introduce the cycle time variable C , we will use it here to simplify the notation (see constraint (A1)). We emphasize that all our modifications leave the formulation equivalent to the original version.

The objective is to minimize the cycle time C , where the cycle time is considered to be the return time of the hoist after the execution of the latest move, see constraint (A1).

$$C = t_{\max} + \sum_{i=1}^n (d_i + e_{i+1,0})z_i \quad (A1)$$

The first set of constraints, (A2)–(A4), ensures that there is exactly one latest move, and that the variable $t_{\max}$ is properly defined.

$$\sum_{i=1}^n z_i = 1 \quad (A2)$$

$$t_{\max} \geq t_i \quad 1 \leq i \leq n \quad (A3)$$

$$t_{\max} \leq t_i + M(1 - z_i) \quad 1 \leq i \leq n \quad (A4)$$

The second set of constraints, (A5)–(A6), consists of the travel time constraints (cf. constraints (3)–(4)). Note that the case $j = i + 1$ for constraint (A6) is covered by the first inequality of constraint (A7).

$$t_j \geq (t_i + d_i) + e_{i+1,j} - M(1 - y_{i,j}) \quad 0 \leq i < j \leq n \quad (A5)$$

$$t_j \geq (t_j + d_j) + e_{j+1,i} - M y_{i,j} \quad 1 \leq i < j \leq n : i + 1 < j \quad (A6)$$

The third set of constraints, (A7)–(A8), consists of the soaking constraints (cf. constraints (9)–(10)).

$$L_i - M(1 - y_{i-1,i}) \leq t_i - (t_{i-1} + d_{i-1}) \leq U_i + M(1 - y_{i-1,i}) \quad 1 \leq i \leq n \quad (A7)$$

$$L_i - M y_{i-1,i} \leq (C + t_i) - (t_{i-1} + d_{i-1}) \leq U_i + M y_{i-1,i} \quad 1 \leq i \leq n \quad (A8)$$

In summary, the formulation of Phillips and Unger (1976) aims to minimize the cycle time C , subject to constraints (A1)–(A8), where $t_0 = 0$, $0 \leq t_i$ ($1 \leq i \leq n$), $y_{0,j} = 1$ ($1 \leq j \leq n$), $y_{i,j} \in \{0, 1\}$ ($1 \leq i < j \leq n$), $z_i \in \{0, 1\}$ ($1 \leq i \leq n$).

4.1.1. Extension to multi-function tanks

Phillips and Unger (1976) extended their formulation to multi-function tanks, by adding constraint (A9) to the model, and by replacing the appropriate constraints (A5)–(A6) with the following ones:

$$y_{i-1,i} + y_{j-1,j} \geq 1 \quad 0 \leq i + 1 < j \leq n : s_i = s_j \quad (A9)$$

$$t_{j-1} \geq t_{i-1} + d_{i-1} + L_i + d_i + e_{i+1,j} - M(1 - y_{i-1,j-1}) \quad 0 \leq i + 1 < j \leq n : s_i = s_j \quad (A10)$$

$$t_{i-1} \geq t_{j-1} + d_{j-1} + L_j + d_j + e_{j+1,i} - M y_{i-1,j-1} \quad 0 \leq i + 1 < j \leq n : s_i = s_j \quad (A11)$$

Constraint (A9) ensures that at least one of the operations is not in process at the beginning of the cycle. Constraints (A10)–(A11) enforce a minimum gap between the operation start times for the corresponding operation end time.

4.1.2. Remarks, issues

In the following, we discuss our observations regarding the formulation.

Unspecified loading and unloading constraints. Although Phillips and Unger (1976) provided an example in which a lower bound was given for operation 0, their formulation did not include any constraints to enforce this. This inconsistency can be easily fixed by adding constraint (11) or (12) to the model, depending on the load-unload configuration.

Narrowed solution space. To the best of our knowledge, this issue has not been raised before. Phillips and Unger (1976) fixed not only the beginning (recall that $t_0 = 0$), but also the end of the cycle. That is, they considered the expression $t_{\max} + \sum_{i=1}^n (d_i + e_{i+1,0})z_i$ as the cycle time, which is the start time of the latest move, plus its execution time, plus the empty travel time to tank 0. Thus, the cycle immediately ends with the return of the hoist to the load station after the execution of the latest move. We call such a formulation *restricted* with respect to the cycle finish. This eliminates the possibility of the hoist waiting at the load station at the end of the cycle to further soak the parts in the tanks. It has several consequences. Clearly, this restricted structure narrows the set of feasible solutions, and therefore the resulting cycle time may be higher than the true optimum. Moreover, if $e_{n+1,0} < L_0$ and constraint (12) is imposed, then the primitive solution constructed

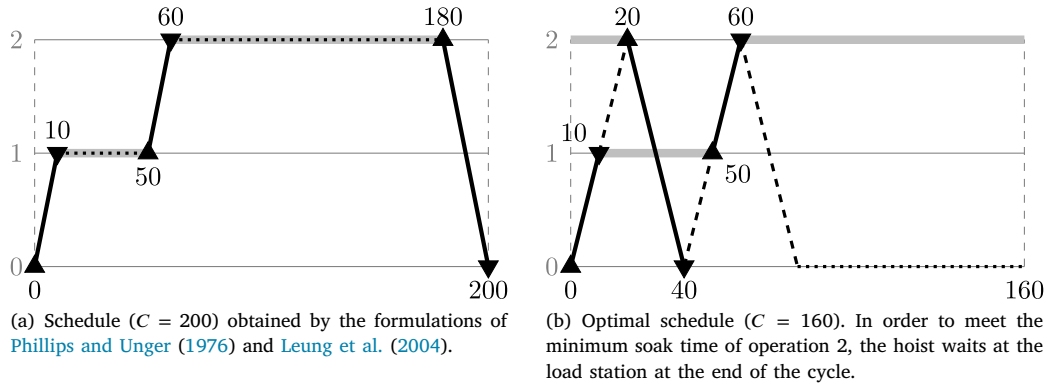


Fig. 6. Instance ex1 shows that formulations that are restricted with respect to the cycle finish may lead to a suboptimal solution.

Table 5
Data for the instance ex1.

i	$e_{i,0}$	$e_{i,1}$	$e_{i,2}$	$e_{i,3}$	s_i	d_i	L_i	U_i
0	0	10	20	0	0	10	0	∞
1	10	0	10	10	1	10	40	100
2	20	10	0	20	2	20	120	∞
3	0	10	20	0	0	–	0	∞

in Section 3.1.1 may no longer be feasible. Consequently, the upper bound (1) may become invalid, and the formulation itself may even be infeasible.

Consider the mini example with two soaking operations indicated in Table 5. Note that the load and unload stations are associated, but loading and unloading constraints are unnecessary as $L_0 = L_{n+1} = 0$ and $U_0 = U_{n+1} = \infty$. There are two possible orders of moves: (0, 1, 2) and (0, 2, 1). In the first case, the minimum cycle time is $d_0 + L_1 + d_1 + L_2 + d_3 = 200$, see Fig. 6(a). In the latter case, the cycle time is at least $d_2 + e_{0,1} + d_1 + L_2 = 160$, which can be reached, see Fig. 6(b). The situation changes when the hoist must immediately return to the load station after executing the latest move, and then the cycle ends. In case of move order (0, 2, 1), move 2 cannot start earlier than $L_2 - e_{2,0} = 100$, but move 1 cannot start later than $d_0 + U_1 = 110$, which is impossible. Thus, for the model of Phillips and Unger (1976), only the first move order is feasible, so it results in a suboptimal solution.

Multi-function tanks. Note that constraints (A9)–(A11) do not prevent operations using the same tank from overlapping. Recall that constraint (A10) aims to enforce that if $t_{i-1} < t_{j-1}$, then $t_i < t_{j-1}$. As Liu et al. (2002) pointed out, this constraint does not prevent the case $t_{i-1} + d_{i-1} < t_j < t_i < t_{j-1} + d_{j-1}$, that is, operation j crosses the cycle boundary and overlaps with operation i . The authors provided an example, in which the model produces an incorrect solution.

We also mention that constraint (A10) may also fail to prevent the case $t_{i-1} < t_{j-1} < t_j$. The constraint only forces a minimum gap between t_{i-1} and t_{j-1} based on the lower bound L_i , however, the operation may have a duration longer than L_i , and thus longer than the enforced gap. Thus, t_i may be greater than t_{j-1} .

4.2. Formulation of Leung et al. (2004)

Leung et al. (2004) revised the model of Phillips and Unger (1976) and proposed a tighter formulation, where two big-M constraints of the former model are simplified, and several valid inequalities are introduced.

We use the cycle time variable C , even though the authors only introduced an auxiliary variable along with the constraint $d = \sum_{i=1}^n (d_i +$

$e_{i+1,0})z_i$. Again, we note that the formulation remains equivalent to the original one.

Constraints (A1)–(A4) are used unchanged, see (B1)–(B4).

$$C = t_{\max} + \sum_{i=1}^n (d_i + e_{i+1,0})z_i \quad (B1)$$

$$\sum_{i=1}^n z_i = 1 \quad (B2)$$

$$t_{\max} \geq t_i \quad 1 \leq i \leq n \quad (B3)$$

$$t_{\max} \leq t_i + M(1 - z_i) \quad 1 \leq i \leq n \quad (B4)$$

Travel time constraints are separated for tank 0 (see constraint (B5)) and for tanks $1, \dots, n$ (see constraints (B6)–(B7)). That is, $y_{0,j}$ is explicitly substituted by 1.

$$t_j \geq d_0 + e_{1,j} \quad 1 \leq j \leq n \quad (B5)$$

$$t_j \geq (t_i + d_i) + e_{i+1,j} - M(1 - y_{i,j}) \quad 1 \leq i < j \leq n \quad (B6)$$

$$t_j \geq (t_j + d_j) + e_{j+1,i} - M y_{i,j} \quad 1 \leq i < j \leq n \quad (B7)$$

Leung et al. (2004) observed that the big-M terms can be omitted from the right-hand-side of the soaking constraint (B9) and from the left-hand-side of the soaking constraint (B10).

$$L_1 \leq t_1 - d_0 \leq U_1 \quad (B8)$$

$$L_i - M(1 - y_{i-1,i}) \leq t_i - (t_{i-1} + d_{i-1}) \leq U_i \quad 2 \leq i \leq n \quad (B9)$$

$$L_i \leq (C + t_i) - (t_{i-1} + d_{i-1}) \leq U_i + M y_{i-1,i} \quad 2 \leq i \leq n \quad (B10)$$

Leung et al. (2004) also proposed valid inequalities to tighten their formulation. These constraints are based on the observation, that the latest move has no successors and all other moves precede it (see constraints (B11) and (B12)), while a non-latest move has at least one successor (see constraint (B13)). Note that the first two inequalities complement each other, that is, the inequality $1 - y_{j,i} \leq 1 - z_i$ ($1 \leq j < i \leq n$) is the same as constraint (B12).

$$y_{i,j} \leq 1 - z_i \quad 1 \leq i < j \leq n \quad (B11)$$

$$z_j \leq y_{i,j} \quad 1 \leq i < j \leq n \quad (B12)$$

$$1 - z_i \leq \sum_{j=1}^{i-1} (1 - y_{j,i}) + \sum_{j=i+1}^n y_{i,j} \quad 1 \leq i \leq n \quad (B13)$$

In summary, the formulation of Leung et al. (2004) aims to minimize C subject to constraints (B1)–(B13), where $t_0 = 0$, $0 \leq t_i$ ($1 \leq i \leq n$), $y_{i,j} \in \{0, 1\}$ ($1 \leq i < j \leq n$), $z_i \in \{0, 1\}$ ($1 \leq i \leq n$).

4.2.1. Remarks, issues

In the following, we discuss our observations regarding the formulation.

Unspecified loading and unloading constraints. Similarly to Phillips and Unger (1976) and Leung et al. (2004) also did not specify the loading and unloading constraints for the associated load and unload stations.

Narrowed solution space. Similarly to Phillips and Unger (1976), the formulation of Leung et al. (2004) is also restricted with respect to the cycle finish.

4.3. Formulation of Zhou and Li (2003)

Constraints (A2)–(A8) are used unchanged, see constraints (C2)–(C8). The cycle time variable C is now a lower bound on the return time, see constraint (C1). Loading and unloading are now explicitly taken into account; constraint (C9) refers to associated load and unload configuration (cf. constraint (12)).

$$C \geq t_{\max} + \sum_{i=1}^n (d_i + e_{i+1,0}) z_i \quad (C1)$$

$$\sum_{i=1}^n z_i = 1 \quad (C2)$$

$$t_{\max} \geq t_i \quad 1 \leq i \leq n \quad (C3)$$

$$t_{\max} \leq t_i + M(1 - z_i) \quad 1 \leq i \leq n \quad (C4)$$

$$t_j \geq (t_i + d_i) + e_{i+1,j} - M(1 - y_{i,j}) \quad 0 \leq i < j \leq n \quad (C5)$$

$$t_i \geq (t_j + d_j) + e_{j+1,i} - M y_{i,j} \quad 1 \leq i < j \leq n : i + 1 < j \quad (C6)$$

$$L_i - M(1 - y_{i-1,i}) \leq t_i \quad (C7)$$

$$-(t_{i-1} + d_{i-1}) \leq U_i + M(1 - y_{i-1,i}) \quad 1 \leq i \leq n \quad (C7)$$

$$L_i - M y_{i-1,i} \leq (C + t_i) \quad (C8)$$

$$-(t_{i-1} + d_{i-1}) \leq U_i + M y_{i-1,i} \quad 1 \leq i \leq n \quad (C8)$$

$$L_0 \leq C - (t_n + d_n) \leq U_0 \quad (C9)$$

In summary, the formulation of Zhou and Li (2003) aims to minimize C subject to constraints (C1)–(C9), where $t_0 = 0$, $0 \leq t_i$ ($1 \leq i \leq n$), $y_{i,j} \in \{0, 1\}$ ($1 \leq i < j \leq n$).

4.3.1. Extension to multi-tank operations

Zhou and Li (2003) extended their formulation to the multi-tank case. For a bottleneck operation i , the corresponding constraints (C7)–(C8) are replaced with constraints (C10)–(C12), (cf. constraint (17)–(18)).

$$L_i - M(1 - y_{i-1,i}) \leq ((m-1)C + t_i) - (t_{i-1} + d_{i-1}) \leq U_i + M(1 - y_{i-1,i}) \quad (C10)$$

$$L_i - M y_{i-1,i} \leq (mC + t_i) - (t_{i-1} + d_{i-1}) \leq U_i + M y_{i-1,i} \quad (C11)$$

$$U_i/m \leq C \leq L_i/(m-1) \quad (C12)$$

4.3.2. Remarks

The formulation of Zhou and Li (2003) is *unrestricted* with respect to the cycle finish, as the authors corrected the calculation of the cycle time with constraint (C1). The authors also specify the loading and unloading constraints for associated load and unload stations.

Unspecified multi-function tank constraints. Zhou and Li (2003) proposed an example with a multi-function tank, however, they did not specify the corresponding constraints.

Invalid lower and upper bounds. Zhou and Li (2003) derived constraint (C12) for the multi-tank case by combining the following lower and upper bounds on the cycle time: $L_i/m \leq C \leq L_i/(m-1)$ and $U_i/m \leq C \leq U_i/(m-1)$. However, the bounds $U_i/m \leq C$ and $C \leq L_i/(m-1)$ are not valid in general. Notice that a feasible solution remains feasible if the operation upper bounds are increased. However, in the presence of

constraint (C12), an otherwise feasible problem may become infeasible if U_i is increased beyond $L_i m/(m-1)$.

4.4. Improved extended formulation

Now, we present an improved extended formulation, which is a combination of the previous models. Similarly to Zhou and Li (2003), we correct the calculation of the cycle time (see constraint (D1)). Similarly to Leung et al. (2004), we use the strengthened soaking constraints (D7)–(D8) and the valid inequalities (D11)–(D12).

The objective is to minimize the cycle time C subject to constraints (D1)–(D12), where $0 \leq t_i$ ($1 \leq i \leq n$), $y_{i,j} \in \{0, 1\}$ ($1 \leq i, j \leq n : i \neq j$), $z_i \in \{0, 1\}$ ($1 \leq i \leq n$).

$$C \geq t_{\max} + \sum_{i=1}^n (d_i + e_{i+1,0}) z_i \quad (D1)$$

$$\sum_{i=1}^n z_i = 1 \quad (D2)$$

$$t_{\max} \geq t_i \quad 1 \leq i \leq n \quad (D3)$$

$$t_{\max} \leq t_i + M(1 - z_i) \quad 1 \leq i \leq n \quad (D4)$$

$$y_{i,j} + y_{j,i} = 1 \quad 0 \leq i < j \leq n \quad (D5)$$

$$t_j \geq t_i + d_i + e_{i+1,j} - M(1 - y_{i,j}) \quad 0 \leq i, j \leq n : i \neq j \quad (D6)$$

$$L_i - M(1 - y_{i-1,i}) \leq t_i - (t_{i-1} + d_{i-1}) \leq U_i \quad 1 \leq i \leq n \quad (D7)$$

$$L_i \leq (C + t_i) - (t_{i-1} + d_{i-1}) \leq U_i + M y_{i-1,i} \quad 1 \leq i \leq n \quad (D8)$$

$$t_0 = 0 \quad (D9)$$

$$y_{0,j} = 1 \quad 1 \leq j \leq n \quad (D10)$$

The following are valid inequalities:

$$y_{i,j} \leq 1 - z_i \quad 1 \leq i, j \leq n : i \neq j \quad (D11)$$

$$1 - z_i \leq \sum_{\substack{j=1 \\ j \neq i}}^n y_{i,j} \quad 1 \leq i \leq n \quad (D12)$$

Depending on the configuration of the load and unload stations, constraint (11) or (12) can be used. In case of multi-function tanks, constraint (13); in case of multi-tank operations, constraints (20) and (21)–(22) are added to the model.

4.5. Formulation of Liu et al. (2002)

Liu et al. (2002) considered the scenario in which the values d_i are only lower bounds on the move times, rather than exact values. For this, the non-negative continuous *slack time variable* d_i^+ measures the slack time between the lower bound d_i and the realized time for move i ($0 \leq i \leq n$). By this, the start time of operation i ($1 \leq i \leq n$) is $t_{i-1} + d_{i-1} + d_{i-1}^+$. Note that fixing $d_i^+ = 0$ for each $0 \leq i \leq n$ results in a base formulation with exact move times.

The objective is to minimize the cycle time C , where the cycle time is long enough for the hoist to return to the load station after executing any move, see constraint (E1).

$$C \geq t_i + d_i + d_i^+ + e_{i+1,0} \quad 1 \leq i \leq n \quad (E1)$$

Constraint (E2) is used only if the load and unload stations are associated. Constraints (E3)–(E4) ensure that the soak time bounds are respected.

$$L_0 \leq C - (t_n + d_n + d_n^+) \leq U_0 \quad \text{if } s_{n+1} = s_0 \quad (E2)$$

$$L_i - M(1 - y_{i-1,i}) \leq t_i - (t_{i-1} + d_{i-1} + d_{i-1}^+) \leq U_i + M(1 - y_{i-1,i}) \quad 1 \leq i \leq n \quad (E3)$$

$$L_i - M y_{i-1,i} \leq (C + t_i) - (t_{i-1} + d_{i-1} + d_{i-1}^+) \leq U_i + M y_{i-1,i} \quad 1 \leq i \leq n \quad (E4)$$

Constraints (E5)–(E7) refer to the travel time constraints.

$$t_j \geq d_0 + d_0^+ + e_{1,j} \quad 1 \leq j \leq n \quad (E5)$$

$$t_j \geq t_i + d_i + d_i^+ + e_{i+1,j} - M(1 - y_{i,j}) \quad 1 \leq i < j \leq n \quad (\text{E6})$$

$$t_i \geq t_j + d_j + d_j^+ + e_{j+1,i} - M y_{i,j} \quad 1 \leq i < j \leq n \quad (\text{E7})$$

In summary, the formulation of Liu et al. (2002) aims to minimize C subject to constraints (E1)–(E7), where $0 \leq t_i$ ($1 \leq i \leq n$), $y_{i,j} \in \{0, 1\}$ ($1 \leq i < j \leq n$), $0 \leq d_i^+$ ($1 \leq i \leq n$).

4.5.1. Extension to multi-function tanks

Liu et al. (2002) extended their model to multi-function tanks with the following constraints (cf. Section 3.2.5).

$$y_{i-1,i} + y_{j-1,j} \geq 1 \quad 0 \leq i+1 < j \leq n : s_i = s_j \quad (\text{E8})$$

$$y_{i,j-1} + (1 - y_{i-1,j}) \geq 3 - (y_{i-1,i} + y_{j-1,j}) \quad 0 \leq i+1 < j \leq n : s_i = s_j \quad (\text{E9})$$

4.5.2. Remarks

Liu et al. (2002) presented a base formulation by neglecting additional variables $t_{\max}$ and z_i . Consequently, the formulation is unrestricted with respect to the cycle finish, see constraint (E1). The authors explicitly specified that the loading constraints are used only for associated load and unload stations, see constraint (E2). The authors corrected the multi-function tank constraints of Phillips and Unger (1976). However, note that constraint (E8) is redundant in the presence of constraint (E9), cf. constraint (13).

4.6. Improved base formulation

Now, we present a strengthened version of the base formulation of Liu et al. (2002), where the stronger soaking constraints are used. The objective is to minimize the cycle time C subject to constraints (F1)–(F7), where $0 \leq t_i$ ($1 \leq i \leq n$), $y_{i,j} \in \{0, 1\}$ ($1 \leq i, j \leq n : i \neq j$).

$$C \geq t_i + d_i + e_{i+1,0} \quad 1 \leq i \leq n \quad (\text{F1})$$

$$t_j \geq t_i + d_i + e_{i+1,j} - M(1 - y_{i,j}) \quad 1 \leq i, j \leq n : i \neq j \quad (\text{F2})$$

$$y_{i,j} + y_{j,i} = 1 \quad 1 \leq i < j \leq n \quad (\text{F3})$$

$$L_i - M(1 - y_{i-1,i}) \leq t_i - (t_{i-1} + d_{i-1}) \leq U_i \quad 1 \leq i \leq n \quad (\text{F4})$$

$$L_i \leq (C + t_i) - (t_{i-1} + d_{i-1}) \leq U_i + M y_{i-1,i} \quad 1 \leq i \leq n \quad (\text{F5})$$

$$t_0 = 0 \quad (\text{F6})$$

$$y_{0,j} = 1 \quad 1 \leq j \leq n \quad (\text{F7})$$

Depending on the configuration of the load and unload stations, constraint (11) or (12) can be used. In case of multi-function tanks, constraint (13); in case of multi-tank operations, constraints (20) and (21)–(22) are added to the model.

5. Computational experiments

In this section, we present the results of our computational experiments to compare the proposed formulations. We evaluate them in three complementary ways: by examining the strength of the MIP formulations via their LP-relaxations (Section 5.2); by testing their performance in solving the MIP models on a single thread (Section 5.3.1); and by assessing their multi-thread performance on 8 threads (Section 5.3.2). These three perspectives allow us to isolate the effect of the formulation itself (LP-relax), the solver heuristics and branching (single-thread MIP), and parallelization (multi-thread MIP) on solution quality and computational efficiency.

Note that the proposed CP formulation solves all considered instances to optimality within short running times, and thus provides a simple and effective baseline for practical use. In contrast, MIP formulations enable a deeper analysis of modelling choices, such as the impact of constraint structure and formulation strength on LP relaxations and solver performance. While the CP models focus on solving the problem efficiently, the MIP formulations also provide a

Table 6

Main properties of the investigated and proposed formulations.

Formulation	Type	Cycle finish	Soaking constraints	Valid inequalities
Phillips	Extended	Restricted	Original	No
Leung	Extended	Restricted	Strengthened	No
Leung ⁺	Extended	Restricted	Strengthened	Yes
Zhou	Extended	Unrestricted	Original	No
Imp1	Extended	Unrestricted	Strengthened	No
Imp1 ⁺	Extended	Unrestricted	Strengthened	Yes
Liu	Base	Unrestricted	Original	No
Imp2	Base	Unrestricted	Strengthened	No

framework for systematically comparing and revisiting the modelling literature. Therefore, the two approaches play complementary roles in our study.

In the following tables, we report only summary results, and refer the reader to the supplementary material for the detailed results. Column *Phillips* refers to the formulation (A1)–(A8) of Phillips and Unger (1976). Column *Leung* refers to the formulation (B1)–(B10) of Leung et al. (2004). In case of *Leung⁺*, the model is also extended with the valid inequalities (B11)–(B13). Column *Zhou* refers to the formulation (C1)–(C9) of Zhou and Li (2003). Column *Imp1* refers to the improved extended formulation (D1)–(D10) proposed in Section 4.4. In case of *Imp⁺*, the model is also extended with the valid inequalities (D11)–(D12). Column *Liu* refers to the formulation (E2)–(E9) of Liu et al. (2002) with exact move times (i.e., $d_i^+ = 0$ is fixed for $0 \leq i \leq n$). Column *Imp2* refers to the improved base formulation (F1)–(F7) proposed in Section 4.6. In each case, the objective is to minimize the cycle time C .

In the comparison, we take into account that the formulations differ in their main properties, summarized in Table 6. Note that the loading and unloading constraints were specified differently (or not specified at all) in the formulations, but all formulations could handle both constraints (11) and (12). In order to make a fair comparison, we generated instances with $L_0 = U_0 = 0$ and $L_{n+1} = U_{n+1} = \infty$, thus loading and unloading constraints were irrelevant.

Setup. The models are implemented in Python (version 3.13.11) using Google OR-Tools (version 9.14.6206). The CP models are solved with OR-Tools CP-SAT, while the MIP models are solved with Gurobi (version 12.0.3) (Gurobi Optimization, LLC, 2026). All experiments were conducted in a cloud environment on an Ubuntu 22.04 system equipped with 16 single-threaded Intel Xeon Cascade Lake cores running at 2.1 GHz.

5.1. Instances, preliminary experiments

In Section 5.1.1, we present the widely-used benchmark instances that can be found in the literature. There, we also explain the inconsistencies between the optimal values reported, see Table 1. In Section 5.1.2, we present the procedure for generating random instances that can challenge the modern CP and MIP solvers.

5.1.1. Benchmark instances

Phillips and Unger (1976) proposed an example (PU) for which they reported a solution with a cycle time of 580. Contrary to their claim, this hoist schedule is not optimal; the true optimal cycle time is 521. The ‘optimal’ cycle time reported by Liu et al. (2025) suggests that either their implementation differs from the intended formulation, or a modified instance was used. Our instance philu corresponds to the one proposed by Phillips and Unger (1976), converted into an equivalent instance with dissociative load and unload stations (see Section 3.1). This modification applies to all the following instances as well.

According to Leung et al. (2004) and Laajili et al. (2021), instance Mini is the truncation of instance PU to the first eight tanks. However, the instance itself is not provided, which may cause inconsistencies. For example, it is not clear whether further adjustment of move times is needed, since Phillips and Unger (1976) defined their original instance with $d_i = e_{i,i+1} + 20$. Our instance `philu_mini` is made without any further adjustment. According to our CP model, the optimal cycle time is 284 for dissociated load and unload stations (i.e., with constraint (11)), and the optimum is 340 for associated load and unload stations (i.e., with constraint (12)).

Shapiro and Nuttle (1988) applied their solution approach on four new test problems, Black Oxide 1 (BO1), Black Oxide 2 (BO2), Zinc (Zn), and Copper (Cu), which were provided by a vendor of wetlines. The data for these examples were not provided in their paper, but later Leung et al. (2004) proposed these or similar instances. The original instances included duplicated tanks, allowing for multi-tank operations, however, the later instances are typically solved only with single-tank operations. This modelling difference leads to discrepancies in the reported cycle times, as the underlying problem structures are not fully equivalent. Our instances `bo1`, `bo2`, `zn`, and `cu` are based on (Laajili et al., 2021), but the time values have been multiplied by 10 to obtain integer values, as the CP-SAT solver operates on integers. We emphasize that this transformation does not change the set of feasible solutions. It can be verified, for example, using our CP model, that the optimal objective value for BO1 changes depending on the configuration of the load and unload stations (dissociated: 281.9; associated: 333.2). We obtained the same cycle times with our implementation of the formulation of Leung et al. (2004). Therefore, we could not reproduce the ‘optimal’ cycle time 299.5 reported in Leung et al. (2004). Since the original implementation is not publicly available, the source of this discrepancy cannot be determined. For the other three instances, there is no difference in these two cases (BO2: 279.3, Cu: 1847.2, Zn: 1743.4). Liu et al. (2025) probably rounded the time values to obtain integer values for CP-SAT, however, we could not reproduce their reported cycle times for these instances. In the lack of their original implementation, the source of this discrepancy cannot be determined.

Manier (1994) proposed two new industrial instances (Ligne 1 and Ligne 2), also available in Manier and Lamrous (2008). Ligne 1 originally included a circulation constraint, therefore there was a separate unloading operation associated with a non-zero lower bound. Laajili et al. (2021) slightly modified this instance, removing the unloading operation and adding its lower bound to the lower bound of the loading operation. Our instances `ligne1` and `ligne2` are the ones provided in Laajili et al. (2021). It can be verified, for example, using our CP model, that the optimal cycle time for Ligne 1 depends on the load-unload configuration (dissociated: 392; associated: 425). In case of Ligne 2, the optimal cycle time is 712 in both cases.

5.1.2. Random instances

Some papers proposed procedures for generating random instances, but without providing the instances themselves. Our preliminary experiments showed that the method of Che et al. (2015) generates instances that can challenge the modern CP and MIP solvers, and therefore suitable for our comparisons. The authors generated instances for a robust, single-hoist cyclic scheduling problem with dissociated load and unload stations with parameters $n \in \{12, 14, 16, 18, 20, 22\}$ and $\mu \in \{1.5, 2.0, 2.5\}$ in the following way: $e_{i,i+1} = 1 + \text{uniform}(0, 4)$ ($0 \leq i \leq n$); $e_{i,j+1} = e_{j+1,i} = \sum_{k=i}^j e_{k,k+1}$ ($1 \leq i < j \leq n$); $d_i = e_{i,i+1} + 12$, $L_i = 40 + \text{uniform}(0, 140)$ and $U_i = \lfloor \mu L_i \rfloor$ ($0 \leq i \leq n$), where $\text{uniform}(a, b)$ refers to a uniformly random integer from $\{a, a+1, \dots, b\}$.

Our preliminary experiments showed that, some of the instances generated in this way are infeasible in the presence of loading and unloading constraints, especially for the formulations of Phillips and Unger (1976) and Leung et al. (2004) that are restricted with respect to the cycle finish. Thus, we slightly modified this procedure by setting $L_0 = 0$ and $U_0 = \infty$. The implemented generation procedure can be

found in our library. For our experiments, we generated 10 instances for each pair of parameters from $n \in \{14, 19, 24\}$ and $\mu \in \{1.5, 2.0, 2.5\}$, resulting in 90 instances. Note that for a fixed n , the k th instances for the different μ parameters differ only in the U_i values, that is, the empty travel time matrix, the move times, and the lower bounds are the same.

5.2. LP-relaxations

We computed the root LP bound for each MIP formulation by relaxing all integer and binary variables to continuous within their original bounds (0–1 for binaries), and disabling presolve (`presolve = Emphasis.OFF`). Tables 7 and 8 indicate the computational results on the benchmark instances, and summarize the average computational results on the randomly generated instances, respectively. Gurobi solved each instance in a split of seconds, thus solution times are not indicated. Recall that comparing the restricted formulations of Phillips and Unger (1976) and Leung et al. (2004) to the others may not be fair, since the former formulations have a narrower feasible solution space.

A note on the operation upper bounds. Recall that in our instance generation procedure, the upper bounds on the soaking times were scaled as $U_i = \mu L_i$. Interestingly, we observed that the optimal value of the LP-relaxation remained unchanged for all tested values of μ , despite the fact that the feasible region of the integer problem was modified. Thus, for these types of instances, the upper bounds U_i had little effect in the relaxation.

Strengthened constraints, valid inequalities. The significant improvement of Leung over Phillips came from the use of stronger soaking constraints. Using additional valid inequalities on top of this had only a tiny effect on LP-relaxation (compare Leung and Leung⁺). The same is observed for the unrestricted formulations. Using stronger soaking constraints strengthened the formulations (compare Zhou and Imp1; Liu and Imp2), however, the valid inequalities did not provide much benefit when strong soaking constraints were already in place (compare Imp1 and Imp1⁺). In case of some industrial instances (`cu`, `zn`), the difference is significant between base and stronger soaking constraints.

Base formulations vs. Extended formulations. The base formulations always provided better or equal LP-relaxations than the corresponding extended formulations. Considering the original soaking constraints, the base formulation (Liu) provided 13.9% stronger LP bounds on average than the extended formulations (Phillips, Zhou). The average improvement was 6.5%, 9.3% and 27.0% on the instances with $n = 14$, 19, and 24, respectively. The improvement on a single instance was up to 63.8%. In the case of strengthened soaking constraints, the base formulation (Imp2) provided 7.1% better LP bounds on average than the extended formulations (Leung, Leung⁺, Imp1, Imp1⁺). The average improvement was 3.2%, 7.3% and 11.0% on the instances with $n = 14$, 19, and 24, respectively. The improvement on a single instance was up to 24.5%.

5.3. Performance tests

For these experiments, we set a time limit of 10 min. Accordingly, the status of an optimization was classified as: *No solution found*, meaning that no solution was found within the time limit; *Feasible*, meaning that a feasible solution was found, but it was either not optimal or its optimality was not proved within the time limit; or *Optimal*, meaning that the optimality of the obtained solution was proved.

In the result tables, we summarize the average computational results on instances with fixed n or μ parameters, and for all instances. For these sets, we indicate the number of instances with solution status Optimal/Feasible/No solution found; the geometric mean of running times in seconds; the average optimality gap, calculated as $100\% \times (\text{UB}^{\text{MIP}} - \text{LB}^{\text{MIP}}) / \text{UB}^{\text{MIP}}$, where UB^{MIP} is the objective value of the best found solution, and LB^{MIP} is the best lower bound; and the average gap

Table 7

LP-relaxation values on the benchmark instances.

Instance	OPT ^a	Formulations							
		Phillips	Leung	Leung ⁺	Zhou	Imp1	Imp1 ⁺	Liu	Imp2
philu	521	211.0	211.0	211.0	211.0	211.0	211.0	217.0	217.0
philu_mini	284	203.0	203.0	203.0	203.0	203.0	203.0	217.0	217.0
cu	18472	2756.4	9316.5	9316.6	2756.0	9316.6	9316.6	2834.0	9404.5
zn	17434	2709.0	8725.5	8725.6	2709.0	8725.5	8725.5	2875.0	8879.0
bo1	2819	2205.0	2205.0	2206.3	2205.5	2205.0	2206.3	2378.0	2378.0
bo2	2793	2205.0	2205.0	2206.5	2205.0	2205.0	2206.5	2378.0	2378.0
ligne1	392	107.0	188.0	188.0	107.0	188.0	188.0	124.0	191.0
ligne2	712	215.0	369.0	369.1	215.0	369.0	369.1	224.0	380.5

^a Obtained with the CP formulation proposed in Section 3.2.9 without loading/unloading constraints.**Table 8**

Average LP-relaxation values on randomly generated instances.

Instance parameters	Formulations							
	Phillips	Leung	Leung ⁺	Zhou	Imp1	Imp1 ⁺	Liu	Imp2
$n = 14, \mu = 1.5$	147.3	165.0	165.1	147.3	165.0	165.1	156.9	170.3
$n = 14, \mu = 2.0$	147.3	165.0	165.1	147.3	165.0	165.1	156.9	170.3
$n = 14, \mu = 2.5$	147.3	165.0	165.1	147.3	165.0	165.1	156.9	170.3
$n = 19, \mu = 1.5$	147.9	156.7	156.7	147.9	156.7	156.7	161.6	168.1
$n = 19, \mu = 2.0$	147.9	156.7	156.7	147.9	156.7	156.7	161.6	168.1
$n = 19, \mu = 2.5$	147.9	156.7	156.7	147.9	156.7	156.7	161.6	168.1
$n = 24, \mu = 1.5$	135.1	157.8	157.9	135.1	157.8	157.9	171.6	175.2
$n = 24, \mu = 2.0$	135.1	157.8	157.9	135.1	157.8	157.9	171.6	175.2
$n = 24, \mu = 2.5$	135.1	157.8	157.9	135.1	157.8	157.9	171.6	175.2
$n = 14$	147.3	165.0	165.1	147.3	165.0	165.1	156.9	170.3
$n = 19$	147.9	156.7	156.7	147.9	156.7	156.7	161.6	168.1
$n = 24$	135.1	157.8	157.9	135.1	157.8	157.9	171.6	175.2
All	143.4	159.8	159.9	143.4	159.8	159.9	163.4	171.2

to the optimum, calculated as $100\% \times (\text{UB}^{\text{MIP}} - \text{OPT}) / \text{UB}^{\text{MIP}}$, where OPT is the optimal solution.

5.3.1. Single thread tests

Table 9 summarizes the results obtained with the Gurobi solver using a single thread (Threads = 1).

Restricted formulations. In three cases, the optimal solution obtained with formulations *Phillips*, *Leung*, and *Leung⁺* had a cycle time longer than the true optimum due to the modelling restrictions discussed in Section 4.1.2. As a result, the reported gap to the optimum is non-zero, even though the solver status is Optimal and the optimality gap is zero.

Strengthened constraints. Interestingly, the advantage of stronger soaking constraints did not persist in the single-thread setting; in fact, in several cases the effect was reversed. Among the restricted formulations, *Leung* solved three fewer instances to optimality than *Phillips*, which also resulted in an increase in the average running time. At the same time, the solutions obtained by *Leung* were often closer to the true optimum, as reflected by the smaller gap to optimality. For the unrestricted extended formulations, *Imp1* solved one fewer instance to optimality than *Zhou*, and among the base formulations, *Imp2* and *Liu* solved the same number of instances to optimality. In both comparisons, however, the use of stronger constraints consistently reduced the average gap to the optimum.

Valid inequalities. The use of valid inequalities clearly facilitated the solution process. Specifically, for the *Leung⁺* formulation, three additional instances were solved to optimality, while *Imp1⁺* benefited by two instances. In both cases, the computational time decreased and the gaps to the true optimum were reduced.

Base formulations vs. Extended formulations. The base formulations consistently outperformed the corresponding extended versions. Both for

the original and the strengthened soaking constraints, the base formulations (*Liu* and *Imp2*, respectively) solved two more instances to optimality than any of the corresponding extended formulations (*Phillips* and *Zhou*, *Imp1⁺*, respectively), while requiring significantly less computation time.

5.3.2. Multiple threads tests

Table 10 summarizes the results obtained with the Gurobi solver using 8 threads (Threads = 8).

Restricted formulations. Similarly to the single-thread experiments, for three cases, the optimal solution obtained with *Phillips*, *Leung*, and *Leung⁺* had a cycle time longer than the true optimum due to modelling restrictions.

Strengthened constraints. The advantage of stronger soaking constraints was visible in the multi-thread setting. Among the restricted formulations, *Leung* solved one more instance to optimality than *Phillips*, although at the cost of a slightly higher running time. For the unrestricted extended formulations, *Imp1* solved one more instance to optimality than *Zhou*. Among the base formulations, *Imp2* solved all instances to optimality. Note that *Liu* also found the optimal solution for all instances, but failed to prove optimality within the time limit in one case, where the remaining optimality gap was still 27.48% after ten minutes.

Valid inequalities. Although the use of valid inequalities in *Leung⁺* did not increase the number of instances solved to optimality, it reduced the average running time, the average optimality gap, and the average gap to optimality. A similar but slightly stronger effect can be observed for *Imp1⁺*. In this case, the addition of valid inequalities allowed one more instance to be solved to optimality, while maintaining essentially the same average running time, and reducing both the optimality gaps and the gaps to optimality.

Table 9

Average computational results on randomly generated instances (Gurobi, single thread).

Instances	Phillips	Leung	Leung ⁺	Zhou	Imp1	Imp1 ⁺	Liu	Imp2
$\mu = 1.5$	30/0/0	30/0/0	30/0/0	30/0/0	30/0/0	30/0/0	30/0/0	30/0/0
	0.6	0.9	0.7	0.7	0.7	0.7	0.3	0.3
	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
	0.75%	0.75%	0.75%	0.00%	0.00%	0.00%	0.00%	0.00%
$\mu = 2.0$	30/0/0	30/0/0	30/0/0	30/0/0	30/0/0	30/0/0	30/0/0	30/0/0
	8.8	8.8	7.3	8.3	8.5	6.6	3.7	3.9
	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
$\mu = 2.5$	24/6/0	21/9/0	24/6/0	23/7/0	22/8/0	24/6/0	26/4/0	26/4/0
	89.5	90.9	70.6	94.1	90.7	65.2	40.9	37.9
	10.47%	12.43%	8.45%	11.68%	12.04%	8.37%	6.32%	6.19%
	1.14%	0.80%	1.00%	1.18%	0.87%	0.67%	0.41%	0.21%
$n = 14$	30/0/0	30/0/0	30/0/0	30/0/0	30/0/0	30/0/0	30/0/0	30/0/0
	4.3	4.7	3.7	4.1	3.5	3.0	2.1	2.2
	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
	0.37%	0.37%	0.37%	0.00%	0.00%	0.00%	0.00%	0.00%
$n = 19$	28/2/0	26/4/0	28/2/0	28/2/0	26/4/0	28/2/0	29/1/0	29/1/0
	9.5	11.8	8.5	12.3	11.5	9.0	4.1	4.0
	2.41%	3.97%	1.96%	2.59%	3.74%	2.04%	1.13%	0.75%
	0.38%	0.26%	0.15%	0.22%	0.14%	0.06%	0.01%	0.01%
$n = 24$	26/4/0	25/5/0	26/4/0	25/5/0	26/4/0	26/4/0	27/3/0	27/3/0
	11.9	12.3	12.3	11.5	12.7	11.7	4.9	5.3
	8.06%	8.46%	6.49%	9.09%	8.30%	6.34%	5.18%	5.44%
	1.14%	0.92%	1.23%	0.96%	0.73%	0.62%	0.40%	0.20%
All	84/6/0	81/9/0	84/6/0	83/7/0	82/8/0	84/6/0	86/4/0	86/4/0
	7.9	8.8	7.3	8.3	8.0	6.8	3.5	3.6
	3.49%	4.14%	2.82%	3.89%	4.01%	2.79%	2.11%	2.06%
	0.63%	0.52%	0.58%	0.39%	0.29%	0.22%	0.14%	0.07%

Number of instances with status Optimal/Feasible/No solution found.

Geometric mean of running times (s).

Average optimality gap, calculated as $100\% \times (UB^{MIP} - LB^{MIP}) / UB^{MIP}$.Average gap to optimal, calculated as $100\% \times (UB^{MIP} - OPT) / UB^{MIP}$.

Base formulations vs. extended formulations. The base formulations proved to be more effective than the extended versions. Considering the original soaking constraints, *Liu* outperformed both *Phillips* and *Zhou* in all respects. Among the formulations using stronger soaking constraints, *Imp2* outperformed *Leung*, *Leung⁺*, *Imp1*, and *Imp1⁺*.

5.4. Conclusions

Overall, the comparison of the MIP formulations reveals clear differences in formulation strength. The strengthened formulations generally provide tighter bounds and improved computational performance, while the base formulations consistently outperform the corresponding extended versions. These results highlight the importance of formulation design, as even structurally similar models can differ substantially in both bound quality and solution efficiency.

6. Public library for cyclic hoist scheduling

We provide a publicly available library for cyclic hoist scheduling at <https://github.com/hmarko89/CyclicHoistScheduling>, with the aim of supporting reproducible research. The library currently includes:

- Single-hoist, simple cycle: a natural CP formulation and the MIP formulations of *Phillips and Unger (1976)*, *Liu et al. (2002)*, *Zhou and Li (2003)* and *Leung et al. (2004)*, as well as our improved formulations.
- Single-hoist, multi-cycles: a natural CP formulation and the MIP formulations of *Zhou et al. (2012)* and *Li and Fung (2014)*.
- Widely used benchmark instances, example instances, and instance generation procedures from the literature.

All MIP formulations are implemented using OR-Tools MathOpt, which allows the instances to be solved with different MIP solvers

(e.g., Gurobi, SCIP, HiGHS, GLPK). Moreover, by exporting the models as LP files, they can be solved with arbitrary MIP solvers. The CP formulations are implemented using OR-Tools CP-SAT.

Future extensions will include additional MIP and CP formulations, custom solution approaches, and variants for the multi-hoist case, further enhancing the utility of the library for comparative and reproducible research in cyclic hoist scheduling.

7. Conclusion and future work

In this paper, we proposed a unified and structured modelling framework for single-hoist cyclic hoist scheduling problems. The framework establishes a consistent notation and modelling approach that enables a clear and systematic treatment of the problem and its extensions. Compared to existing formulations in the literature, our approach provides a more transparent and coherent representation of the core constraints, while also supporting a broader range of features, including multi-function tanks, multi-degree cycles, and multi-tank operations.

Building on this framework, we introduced natural CP formulations that capture the essential structure of the problem in a transparent and reproducible way. These models can serve as baseline formulations for both simple and extended problem variants.

We revisited MIP formulations from the literature and derived improved variants that integrate their main advantages. Our computational study demonstrated the effectiveness of the proposed formulations, highlighting the impact of modelling choices on both relaxation strength and solver performance.

To support future research, we provided a publicly available library containing benchmark instances and implementations of several CP and MIP formulations. This resource, together with the unified modelling framework, is intended to serve as a common reference point for future

Table 10

Average computational results on randomly generated instances (Gurobi, 8 threads).

Instances	Phillips	Leung	Leung ⁺	Zhou	Imp1	Imp1 ⁺	Liu	Imp2
$\mu = 1.5$	30/0/0	30/0/0	30/0/0	30/0/0	30/0/0	30/0/0	30/0/0	30/0/0
	0.4	0.4	0.4	0.4	0.4	0.4	0.2	0.2
	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
	0.75%	0.75%	0.75%	0.00%	0.00%	0.00%	0.00%	0.00%
$\mu = 2.0$	30/0/0	30/0/0	30/0/0	30/0/0	30/0/0	30/0/0	30/0/0	30/0/0
	3.0	3.4	3.1	3.6	2.9	3.3	1.9	1.7
	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
$\mu = 2.5$	27/3/0	28/2/0	28/2/0	27/3/0	28/2/0	29/1/0	29/1/0	30/0/0
	27.8	26.6	19.8	24.6	28.1	17.7	12.0	11.5
	3.78%	3.16%	2.02%	3.31%	3.34%	1.59%	0.92%	0.00%
	0.40%	0.27%	0.04%	0.01%	0.04%	0.01%	0.00%	0.00%
$n = 14$	30/0/0	30/0/0	30/0/0	30/0/0	30/0/0	30/0/0	30/0/0	30/0/0
	1.5	1.5	1.3	1.3	1.3	1.3	0.8	0.9
	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
	0.37%	0.37%	0.37%	0.00%	0.00%	0.00%	0.00%	0.00%
$n = 19$	30/0/0	30/0/0	30/0/0	30/0/0	30/0/0	30/0/0	30/0/0	30/0/0
	3.9	4.3	3.5	3.8	3.9	3.3	1.9	1.8
	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
	0.12%	0.12%	0.12%	0.00%	0.00%	0.00%	0.00%	0.00%
$n = 24$	27/3/0	28/2/0	28/2/0	27/3/0	28/2/0	29/1/0	29/1/0	30/0/0
	5.1	5.4	5.3	6.4	5.8	5.9	2.6	2.7
	3.78%	3.16%	2.02%	3.31%	3.34%	1.59%	0.92%	0.00%
	0.66%	0.53%	0.30%	0.01%	0.04%	0.01%	0.00%	0.00%
All	87/3/0	88/2/0	88/2/0	87/3/0	88/2/0	89/1/0	89/1/0	90/0/0
	3.1	3.3	2.9	3.2	3.0	3.0	1.6	1.6
	1.26%	1.05%	0.67%	1.10%	1.11%	0.53%	0.31%	0.00%
	0.38%	0.34%	0.26%	0.00%	0.01%	0.00%	0.00%	0.00%

Number of instances with status Optimal/Feasible/No solution found.

Geometric mean of running times (s).

Average optimality gap, calculated as $100\% \times (UB^{MIP} - LB^{MIP}) / UB^{MIP}$.Average gap to optimal, calculated as $100\% \times (UB^{MIP} - OPT) / UB^{MIP}$.

studies and as a baseline for the development and evaluation of new solution approaches.

Ongoing work focuses on extending the proposed modelling framework to the multi-hoist cyclic scheduling problem, where additional challenges arise from the interaction and coordination of multiple hoists. Furthermore, we plan to expand the public library with additional formulations, benchmark instances, and solution methods, including both exact and heuristic approaches.

Declaration of generative AI and AI-assisted technologies in the manuscript preparation process

During the preparation of this work, the author used *ChatGPT (OpenAI)* in order to improve language clarity and phrasing, and *Scholar Labs* in order to explore related literature and identify relevant references. After using these tools, the author reviewed and edited the content as needed and takes full responsibility for the content of the published article.

Acknowledgements

This project has received funding from the European Union's Horizon Research and Innovation Actions programme under grant agreement No. 101137954. The author would like to thank the rest of the BATTwin consortium for supporting this research. The author is grateful for the possibility to use HUN-REN Cloud (see Héder et al. (2022); <https://science-cloud.hu/>) which helped achieve the results published in this paper.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at <https://doi.org/10.1016/j.cor.2026.107600>.

Data availability

Data will be made available on request.

References

- Armstrong, R., Lei, L., Gu, S., 1994. A bounding scheme for deriving the minimal cycle time of a single-transporter N-stage process with time-window constraints. *European J. Oper. Res.* 78 (1), 130–140.
- Che, A., Chu, C., 2004. Single-track multi-hoist scheduling problem: A collision-free resolution based on a branch-and-bound approach. *Int. J. Prod. Res.* 42 (12), 2435–2456.
- Che, A., Chu, C., Chu, F., 2002. Multicyclic hoist scheduling with constant processing times. *IEEE Trans. Robot. Autom.* 18 (1), 69–80.
- Che, A., Feng, J., Chen, H., Chu, C., 2015. Robust optimization for the cyclic hoist scheduling problem. *European J. Oper. Res.* 240 (3), 627–636.
- Che, A., Lei, W., Feng, J., Chu, C., 2013. An improved mixed integer programming approach for multi-hoist cyclic scheduling problem. *IEEE Trans. Autom. Sci. Eng.* 11 (1), 302–309.
- Che, A., Yan, P., Yang, N., Chu, C., 2010. Optimal cyclic scheduling of a hoist and multi-type parts with fixed processing times. *Int. J. Prod. Res.* 48 (5), 1225–1243.
- Che, A., Zhou, Z., Chu, C., Chen, H., 2011. Multi-degree cyclic hoist scheduling with time window constraints. *Int. J. Prod. Res.* 49 (19), 5679–5693.
- Chen, H., Chu, C., Proth, J.-M., 1998. Cyclic scheduling of a hoist with time window constraints. *IEEE Trans. Robot. Autom.* 14 (1), 144–152.
- Chtourou, S., Manier, M.-A., et al., 2013. A hybrid algorithm for the cyclic hoist scheduling problem with two transportation resources. *Comput. Ind. Eng.* 65 (3), 426–437.
- Dawande, M., Geismar, H.N., Sethi, S.P., Sriskandarajah, C., 2005. Sequencing and scheduling in robotic cells: Recent developments. *J. Sched.* 8 (5), 387–426.
- El Amraoui, A., Benrejeb, M., 2024. A new heuristic optimization approach to the single hoist cyclic scheduling problem. *Eng. Technol. Appl. Sci. Res.* 14 (6), 18767–18773.
- El Amraoui, A., Elhafi, M., 2016. An efficient new heuristic for the hoist scheduling problem. *Comput. Oper. Res.* 67, 184–192.
- El Amraoui, A., Manier, M.-A., El Moudni, A., Benrejeb, M., 2008. A mixed linear program for a multi-part cyclic hoist scheduling problem. *Int. J. Sci. Tech. Autom. Control. Comput. Eng. (IJ-STA), Spec. Issue CEM 2*, 612–623.

- El Amraoui, A., Manier, M.-A., El Moudni, A., Benrejeb, M., 2012. Resolution of the two-part cyclic hoist scheduling problem with bounded processing times in complex lines' configuration. *Eur. J. Ind. Eng.* 6 (4), 454–473.
- El Amraoui, A., Manier, M.-A., El Moudni, A., Benrejeb, M., 2013. A linear optimization approach to the heterogeneous r-cyclic hoist scheduling problem. *Comput. Ind. Eng.* 65 (3), 360–369.
- El Amraoui, A., Nait-Sidi-Moh, A., 2012. P-temporal petri nets for hoist scheduling problem. *IFAC Proc.* Vol. 45 (6), 297–302.
- Feng, J., 2017. Modélisation et optimisation des Hoist Scheduling Problems (Ph.D. thesis). Université Paris Saclay (COMUE); Northwestern Polytechnical University (X ...).
- Feng, J., Che, A., Chu, C., 2015. Dynamic hoist scheduling problem with multi-capacity reentrant machines: A mixed integer programming approach. *Comput. Ind. Eng.* 87, 611–620.
- Feng, J., Che, A., Chu, C., Levner, E., Kats, V., 2024. Scheduling robotic cells with fixed processing times or time windows: Classification, solution approaches, polynomial algorithms and complexity. *European J. Oper. Res.* 319 (2), 468–483.
- Feng, J., Che, A., Wang, N., 2014. Bi-objective cyclic scheduling in a robotic cell with processing time windows and non-Euclidean travel times. *Int. J. Prod. Res.* 52 (9), 2505–2518.
- Feng, J., Chu, C., Che, A., 2018. Cyclic jobshop hoist scheduling with multi-capacity reentrant tanks and time-window constraints. *Comput. Ind. Eng.* 120, 382–391.
- Graham, R.L., Lawler, E.L., Lenstra, J.K., Kan, A.R., 1979. Optimization and approximation in deterministic sequencing and scheduling: A survey. In: *Annals of Discrete Mathematics*. vol. 5, Elsevier, pp. 287–326.
- Grötschel, M., Jünger, M., Reinelt, G., 1985. Facets of the linear ordering polytope. *Math. Program.* 33 (1), 43–60.
- Gurobi Optimization, LLC, 2026. Gurobi optimizer reference manual. URL <https://www.gurobi.com>.
- Héder, M., Rigó, E., Medgyesi, D., Lovas, R., Tenczer, S., Farkas, A., Emödi, M.B., Kadlecisk, J., Kacsuk, P., 2022. The past, present and future of the ELKH cloud. *Inform. Társadalom: Társadalomtudományi Folyóirat* 22 (2), 128–137.
- Jégou, D., Kim, D.-W., Baptiste, P., Lee, K.H., 2006. A contract net based intelligent agent system for solving the reactive hoist scheduling problem. *Expert Syst. Appl.* 30 (2), 156–167.
- Laajili, E., 2021. Modélisation et Algorithmes Pour Le Dimensionnement Et L'ordonnement Cyclique D'atelier De Traitement De Surface (Ph.D. thesis). Université Bourgogne Franche-Comté, in French.
- Laajili, E., Lamrous, S., Manier, M.-A., Nicod, J.-M., 2021. An adapted variable neighborhood search based algorithm for the cyclic multi-hoist design and scheduling problem. *Comput. Ind. Eng.* 157, 107225.
- Lei, W., Che, A., Chu, C., 2014. Optimal cyclic scheduling of a robotic flowshop with multiple part types and flexible processing times. *Eur. J. Ind. Eng.* 8 (2), 143–167.
- Leung, J.M., Levner, E., 2006. An efficient algorithm for multi-hoist cyclic scheduling with fixed processing times. *Oper. Res. Lett.* 34 (4), 465–472.
- Leung, J.M., Zhang, G., Yang, X., Mak, R., Lam, K., 2004. Optimal cyclic multi-hoist scheduling: A mixed integer programming approach. *Oper. Res.* 52 (6), 965–976.
- Levner, E., Kats, V., De Pablo, D.A.L., Cheng, T.E., 2010. Complexity of cyclic scheduling problems: A state-of-the-art survey. *Comput. Ind. Eng.* 59 (2), 352–361.
- Levner, E., Kats, V., Levit, V.E., 1997. An improved algorithm for cyclic flowshop scheduling in a robotic cell. *European J. Oper. Res.* 97 (3), 500–508.
- Li, X., Fung, R.Y., 2014. A mixed integer linear programming solution for single hoist multi-degree cyclic scheduling with reentrance. *Eng. Optim.* 46 (5), 704–723.
- Liu, J., Jiang, Y., Zhou, Z., 2002. Cyclic scheduling of a single hoist in extended electroplating lines: A comprehensive integer programming solution. *Iie Trans.* 34 (10), 905–914.
- Liu, S., Ma, R., Hu, X., Yang, Y., 2025. A fast solving method for the single-hoist cyclic scheduling problem based on google OR-tools. In: 2025 6th International Conference on Machine Learning and Computer Application. ICMLCA, IEEE, pp. 775–779.
- Manier, M.-A., 1994. Contribution à l'Ordonnement Cyclique Du Système De Manutention D'une Ligne De Galvanoplastie (Ph.D. thesis). Besançon.
- Manier, M.-A., Bloch, C., 2003. A classification for hoist scheduling problems. *Int. J. Flexible Manuf. Syst.* 15 (1), 37–55.
- Manier, M.-A., Lamrous, S., 2008. An evolutionary approach for the design and scheduling of electroplating facilities. *J. Math. Model. Algorithms* 7 (2), 197–215.
- Mao, Y.-n., Tang, Q.-h., Li, Z.-x., Zhang, L.-p., 2018. Mixed-integer linear programming method for multi-degree and multi-hoist cyclic scheduling with time windows. *Eng. Optim.* 50 (11), 1978–1995.
- Nait-Sidi-Moh, A., El-Amraoui, A., 2016. Modeling and optimization of cyclic hoist schedules in an electroplating line. *J. Syst. Sci. Syst. Eng.* 25 (4), 469–490.
- Ng, W., 1995. Determining the optimal number of duplicated process tanks in a single-hoist circuit board production line. *Comput. Ind. Eng.* 28 (4), 681–688.
- Phillips, L.W., Unger, P.S., 1976. Mathematical programming solution of a hoist scheduling program. *AIIE Trans.* 8 (2), 219–225.
- Riera, D., Yorke-Smith, N., 2002. An improved hybrid model for the generic hoist scheduling problem. *Ann. Oper. Res.* 115 (1), 173–191.
- Shapiro, G.W., Nuttle, H.L., 1988. Hoist scheduling for a PCB electroplating facility. *IIE Trans.* 20 (2), 157–167.
- Wallace, M., Yorke-Smith, N., 2020. A new constraint programming model and solving for the cyclic hoist scheduling problem. *Constraints* 25 (3), 319–337.
- Xu, Q., Huang, Y., 2004. Graph-assisted cyclic hoist scheduling for environmentally benign electroplating. *Ind. Eng. Chem. Res.* 43 (26), 8307–8316.
- Yan, P., Che, A., Levner, E., Liu, S.Q., 2017. A heuristic for inserting randomly arriving jobs into an existing hoist schedule. *IEEE Trans. Autom. Sci. Eng.* 15 (3), 1423–1430.
- Yan, P., Chu, C., Yang, N., Che, A., 2010. A branch and bound algorithm for optimal cyclic scheduling in a robotic cell with processing time windows. *Int. J. Prod. Res.* 48 (21), 6461–6480.
- Yan, P., Wang, G., Che, A., Li, Y., 2016. Hybrid discrete differential evolution algorithm for biobjective cyclic hoist scheduling with reentrance. *Comput. Oper. Res.* 76, 155–166.
- Zhao, C., Fu, J., Xu, Q., 2013. Production-ratio oriented optimization for multi-recipe material handling via simultaneous hoist scheduling and production line arrangement. *Comput. Chem. Eng.* 50, 28–38.
- Zhou, Z., Che, A., Yan, P., 2012. A mixed integer programming approach for multi-cyclic robotic flowshop scheduling with time window constraints. *Appl. Math. Model.* 36 (8), 3621–3629.
- Zhou, Z., Li, L., 2003. Single hoist cyclic scheduling with multiple tanks: a material handling solution. *Comput. Oper. Res.* 30 (6), 811–819.