IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 6, 2020, accepted May 18, 2020, date of publication May 26, 2020, date of current version June 4, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2997739

Reclaimer Scheduling in Dry Bulk Terminals

ÖZGÜR ÜNSAL
Graduate School of Sciences and Engineering, Koç University, 34450 Istanbul, Turkey
e-mail: cunsal@ku.edu.tr

This work was supported in part by the Scientific and Technological Research Council of Turkey under Grant 113M486.

ABSTRACT This paper studies a complex parallel scheduling problem with non-crossing constraint, sequence dependent setup times, eligibility restrictions, and precedence relationships motivated by reclaimer scheduling in dry bulk terminals. In a stockyard of any dry bulk terminal, stockpiles are handled by reclaimers. Therefore, improving the operational efficiency of reclaimers is critical for the overall performance of these terminals which are struggling with increasing workload. We study the variants of this problem with and without stacking operations. For each variant, we present a lower and an upper bound. A strong lower bound is obtained by relaxing the non-crossing constraint and solving the resulting problem to the optimality. While this relaxation still addresses a challenging scheduling problem, proposed arc-time-indexed formulation copes with the instances of practical size. We develop a novel constraint programming formulation to provide an upper bound for the problem. Computational experiments show this robust approach is able to generate near-optimal schedules for different stockyard configurations within a minute.

INDEX TERMS Parallel machines scheduling, mathematical programming, reclaimer scheduling, bulk terminals.

I. INTRODUCTION

Bulk terminals are seaside facilities in which agricultural products and natural resources are stored and transshipped in very large volumes. Most of the cargo handled in these terminals are major import and export products such as coal, grains, and iron ore. For this reason, bulk loads consist of 83% of all sea-freight in volume terms [1]. Despite the critical role of bulk terminals on global trade, operational problems of these terminals are rarely studied in the literature until recently, in contrast to those of container terminals [2].

An ever increasing workload entails bulk terminals to improve their operational efficiency. Such improvements would benefit both stakeholders as the throughput of a terminal increases and the turnaround time of vessels decreases. Accordingly, in this paper, we address scheduling of reclaimers in dry bulk terminals. In these terminals, cargo is stored as a stockpile and is handled by automated or semi-automated reclaimer machines, thus scheduling of reclaimers is one of the key factors affecting overall performance of a dry bulk terminal [3].

A reclaimer is a huge equipment that reclaims (i.e., collects, recovers) stockpiles from the stockyard to be loaded to mooring vessels. A stockpile is then transferred to a vessel via a conveyor belt system. In the stockyard of a dry bulk terminal, there are parallel stocking pads where stockpiles of different lengths are stacked. In between each two stocking pads, there is a rail track on which one or more reclaimers are mounted. Figure 1 depicts a stockyard with three stocking pads and three reclaimers, where dashed rectangles represent stockpiles.

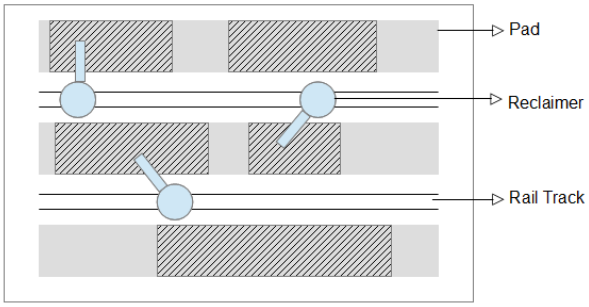


FIGURE 1. Overview of a stockyard of a dry bulk terminal.

Reclaimer scheduling problem (RSP) is to find a schedule for reclaiming stockpiles which are stacked in a stockyard by using a set of reclaimers with respect to the various problem constraints. Reclaimers are identical, and a reclaimer can

The associate editor coordinating the review of this manuscript and approving it for publication was Md Asaduzzaman.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

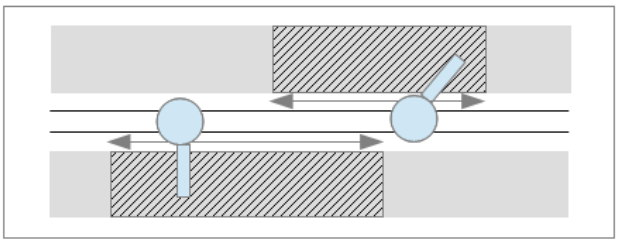


FIGURE 2. Illustration of spatial restriction caused by long travel bench cuts.

only process one task at a time. Reclaiming of a stockpile is assumed to be non-preemptive. Therefore, this problem is a variant of so-called parallel machine scheduling problem (PMSP) with the following properties:

- After finishing the reclaiming of a stockpile, a reclaimer travels along the rail track to go to the area where its next scheduled task is stacked (sequence dependent setup times [4]).
- A reclaimer can rotate its boom to serve the pads that are adjacent to the rail track on which it is mounted, but it cannot reclaim the stockpiles that are located on other pads (eligibility restriction [5]).
- Advanced reclaimers can also perform the stacking of stockpiles. For such machines, if stacking operation is ignored, we might generate infeasible reclaimer schedules as stockpiles are often stacked and reclaimed dynamically during the planning horizon. Therefore, a stockpile must be stacked by a reclaimer before its reclaiming starts (precedence relationships).
- If there are multiple reclaimers on a rail track, they cannot cross each other (non-crossing constraint [6]).

Even though non-crossing constraint is widely investigated in the context of quay crane scheduling problem [7] (QCSP), there are some notable differences between these problems. In QCSP, all QCs operate on a single rail track while reclaimers are distributed across multiple rail tracks. While handling a container, a QC does not move along the rail track. On the other hand, a reclaimer travels along the rail track from one end of the stockpile to the other hand many times to reclaim a stockpile completely (i.e., long travel bench cuts). This brings another spatial restriction for reclaimers located on the same rail track. Consider two such reclaimers and two stockpiles which are located on opposite sides of a rail track. If these two stockpiles are overlapping in time and x-axis, they cannot be reclaimed simultaneously by these two reclaimers since it will cause reclaimers to cross each other (Figure 2).

As QCSP and RSP are related problems as both are the variants of PMSP with non-crossing constraint, the complexity of RSP can be derived by utilizing this relation. That is, if we assume that there is only one pad, one rail track, two reclaimers, and all stockpiles has a unit length, we obtain an instance of QCAP with two QCs. The resulting problem is strongly NP-hard [8].

In this paper, we study RSP by considering every realistic feature of the problem; thus, it is more practical for the application in real-world terminals. Contributions of this paper are two fold.

- We develop an arc-time-indexed mixed integer programming formulation as a relaxation of the considered problem. This formulation provides a strong lower bound by solving the PMSP with sequence dependent setup times and eligibility restrictions to the optimality. Therefore, this formulation not only allows the accurate evaluation of non-exact solution approaches for RSP, but also provides an effective solution to a challenging scheduling problem with a practical relevance.
- To provide an upper bound for the problem, we propose a novel constraint programming formulation. This robust formulation generates near optimal schedules for different stockyard configurations within a minute.

The rest of the paper is organized as follows. Section II presents a concise review of the related literature. In Section III, we first discuss the mixed integer programming formulation of RSP. As it can only solve small sized instances, we develop a mixed integer programming formulation for the relaxed problem and a constraint programming formulation as a lower and an upper bound, respectively. In Section IV, we perform a set of computational experiments to test the proposed methods for different stockyard configurations. Section V concludes the paper with discussion of future research directions.

II. RELATED WORK

The literature on RSP is relatively new. Consequently, this problem is only addressed by only a few papers so far.

Hu and Yao [9] propose a mixed integer programming formulation and a genetic algorithm for RSP in which there is only one reclaimer in each rail track. They assume that a reclaimer can only reclaim stockpiles located in pads adjacent to its rail track and take travel times of reclaimers into account. The similar problem setting is also studied by Wang *et al.* [10]. They develop an approximation algorithm with 2 worst case ratio. They also show that this ratio can be reduced to 3/2 by assuming a task can be executed by two reclaimers simultaneously.

van Vianen *et al.* [11] and Xin *et al.* [12] investigate stockyard operations considering in a realistic setting. However, their focus is on the design of stockyard and allocation of stockpiles that increases throughput, rather than reclaimer scheduling operations.

Angelelli *et al.* [13] introduce an abstract scheduling problem inspired by the reclaiming operations. They present complexity results and an approximation algorithm for a specific setting with two reclaimers mounted on a rail track which is located in-between two pads. This abstract problem has one critical simplification: a reclaimer must travel from the one end of a stockpile to the other end once with a constant speed to reclaim this stockpile completely. However, this is

inconsistent with the real world terminal operations where reclaiming is usually performed by long travel bench cuts. Kalinowski *et al.* [14] extend this abstract problem by relaxing the assumption that requires all stockpiles to be stacked at the beginning of planning period. That is, they study the dynamic version of the problem.

Recently, RSP is studied as a part of more compherensive integrated problems of dry bulk terminals and also coal export supply chains ([15]–[17]). As they study complex problems, unfortunately, they have some simplifications on key characteristics of reclaimer operations such as assuming each rail track has a single reclaimer. An integrated problem considering non-crossing of reclaimers are studied by Unsal and Oguz [18]. They assume that reclaimers are performing long travel bench cuts and take resulting restrictions on the movements of reclaimers into account. However, they ignore travel times of reclaimers as well as stacking operations. Our paper goes beyond RSP literature as it studies the problem with all features observed in real world terminals; specifically, a realistic stockyard structure consists of multiple pads, multiple reclaimers on a rail track, eligibility restrictions, non-crossing constraint with the further restrictions addressed in [18], travel times of reclaimers, and stacking operations.

The most important feature that distinguishes RSP from classical PMSP literature is the spatial restrictions on the movements of reclaimers, namely non-crossing constraint. This constraint is commonly observed in container logistics in maritime terminals as well as warehousing operations in automated storage and retrieval systems where cranes are operating on a same rail track [6]. Numerous studies on crane scheduling including Bierwirth and Meisel [19] and Chen *et al.* [20] show that near optimal (or even optimal in some cases) solutions can be quickly derived assuming so-called unidirectional schedules. However, as we explained in Section I, RSP differs from crane scheduling in terms of non crossing constraint and unidirectional schedules are not applicable for RSP.

As a lower bound for RSP, we simply relax non-crossing constraint and exactly solve the resulting PMSP with sequence dependent setup times and eligibility restrictions. While there are myriad of studies that examine these two extensions individually, to the best of our knowledge, the only exact method for this specific setting of PMSP is proposed by Gokhale and Mathirajan [21]. Note that the formulation we develop strictly outperforms their MIP formulation which embeds eligibility restrictions into processing time parameter (see Section III-B1).

III. METHODOLOGY

A. MIXED INTEGER PROGRAMMING FORMULATION

We initially formulate a mixed integer programming (MIP) formulation based on a network flow variable $X_{i_1 i_2 k}$, which takes a value of 1 if stockpile i_1 is processed immediately after stockpile i_1 on reclaimer k, 0 otherwise. The performance of the MIP formulations of parallel machine scheduling problem with network variables are shown to be ineffective compared to the performance of time-indexed formulations as long as the length of planning horizon is not large ([22], [23]). This can be mainly attributed to its weak linear programming relaxation, and the resulting RSP formulation is not an exception. On the other hand, a time-indexed formulation is impractical for RSP because of sequence dependent setup times and non-crossing constraint. We do not present the constraints of this formulation, since it neither can solve problem instances of reasonable sizes nor will be utilized within the proposed methods. In the following, we list sets and parameters that we use throughout the paper.

J set of reclaiming tasks (stockpiles), $i = \{1, \ldots, |J|\}$,
C set of reclaimers, $k = \{1, \ldots, |C|\}$,
T set of time indices, $t = \{1, \ldots, |T|\}$,
A set of pads in stockyard, $a = \{1, \ldots, |A|\}$,
W set of rail tracks, $w = \{1, \ldots, |W|\}$,
U set of stock positions located on a pad, $u = \{1, \ldots, |U|\}$,
q_k index of a rail track reclaimer k is mounted, $q_k \in W$,
p_i reclaiming time of stockpile i,
k_i number of stock positions required to accommodate stockpile i,
$s_{i_1 i_2}$ travel time between midpoints of stockpiles i_1 and i_2,
pad_i index of pad in which stockpile i is stacked, $pad_i \in A$,
pos_i index of stock position in which leftmost of stockpile i is positioned,
w_i weight associated with stockpile i.

For the problem with stacking operations, we introduce the following additional sets and parameters.

J' extended set of tasks, $i = \{1, \ldots, |J|, |J|+1, \ldots 2|J|\}$, where tasks indexed from $|J|+1$ to $2|J|$ stand for the stacking operations of stockpiles,
δ a constant that defines the estimated ratio of the reclaiming time to the stacking time of a stockpile,
p'_i stacking and reclaiming times of stockpiles, i.e.,

$$p'_i = \begin{cases} p_i, & \text{if } i = \{1, \ldots, |J|\}. \\ \lceil p_i/\delta \rceil, & \text{if } i = \{|J|+1, \ldots 2|J|\}. \end{cases}$$

w'_i weight of each task, i.e.,

$$w'_i = \begin{cases} w_i, & \text{if } i = \{1, \ldots, |J|\}. \\ 0, & \text{if } i = \{|J|+1, \ldots 2|J|\}. \end{cases}$$

γ set of precedence relationships between stacking and reclaiming tasks, $\gamma = \{(i_1, i_2) | i_1 \in \{|J|+1, \ldots, 2|J|\}, i_2 \in \{1, \ldots, |J|\} : i_1 = i_2 + |J|\}$.

Given these sets and parameters the problem we study can be formally defined as follows. Suppose there are $|J|$ tasks ($i \in \{1, \ldots, |J|\}$) to be scheduled on $|C|$ identical parallel machines ($k \in \{1, \ldots, |C|\}$). Each task must be non-preemptively performed by a single machine. Then, RSP is to minimize total weighted completion time of tasks subject to following constraints:

- Machine k is eligible to perform task i if and only if $pad_i = q_k$ or $pad_i = q_k + 1$.

- If machine k performs tasks i_2 immediately after completing task i_1, then a sequence dependent setup time $s_{i_1 i_2}$ incurs.
- If tasks i_1 and i_2 are performed by respective machines k_1 and k_2 where $q_{k_1} = q_{k_2}$, $k_1 < k_2$, and $pos_{i_1} + len_{i_1} > pos_{i_2}$, then these two tasks cannot be performed simultaneously because of the spatial restrictions.

For the variant of the problem with stacking operations, there are $|J|$ additional tasks indexed from $|J|+1$ to 2, which are also subject to the above constraints. In addition, we have the following constraint.

- Task i cannot be performed before the completion of task $|J|+i$.

B. LOWER BOUND FOR RSP: MIP FORMULATION

We develop an arc-time-indexed MIP formulation that provides a strong lower bound for RSP to evaluate the performance of non-exact solution methods accurately. This type of formulation is first presented for single machine by Sourd [24] and is extended to parallel machines by Pessoa *et al.* [25]. Arc-time-indexed formulations integrate respective advantages of network and time-indexed formulations: the former keeps a sequence of tasks for each machine while the latter provides a strong LP relaxation as it does not need "big M" to model disjunctions. In the proposed arc-time-indexed formulations that we study the lower bound for RSP with and without stacking operations separately, we simply relax non-crossing constraint but keep all other features of the problem.

1) LOWER BOUND FOR RSP WITHOUT STACKING OPERATIONS

Differently from [25], we add sequence dependent setup times to arc-time-indexed formulation, introduce machine index to model eligibility restrictions, and also eliminate the variables that address the idleness of machines with the following observation.

Definition 1: A feasible schedule is called **non-delay**, if no machine is kept idle, while there is at least one task waiting for further processing.

Proposition 1: Assuming a regular objective, there exist an optimal solution in which each machine has a non-delay schedule.

Proof: In any optimal solution, tasks are assigned to machines with respect to eligibility restrictions. Given such an assignment, the problem can be decomposed into $|C|$ independent single machines as we relax the non-crossing constraint. For a single machine scheduling problem with sequence dependent setup times, there exist an optimal schedule which is non-delay [26]. □

Accordingly, we introduce a binary variable $Y_{i_1 i_2 kt}$ that takes a value of 1 if machine k starts processing task i_2 at time t, immediately after processing task i_1, 0 otherwise. In this formulation, for each machine, we determine a route over nodes where a node represents a task-time tuple (i, t), rather than a task as in pure network flow formulation. We also define a dummy task, indexed by 0, with null processing time and introduce a new set J^0, set of tasks extended with dummy task 0 ($J^0 = J \cup 0$). This MIP model for relaxed RSP is presented below.

$$[RSP - LB]:$$

$$\text{minimize} \sum_{i_1 \in J^0} \sum_{i_2 \in J} \sum_{k} \sum_{t} (t + p_{i_2}) w_{i_2} Y_{i_1 i_2 kt}$$

subject to:

$$\sum_{i_1 \in J^0} \sum_{k} \sum_{t} Y_{i_1 i_2 kt} = 1 \quad \forall i_2 \in J \tag{1}$$

$$\sum_{i_1 \in J^0} \sum_{t} Y_{i_1 i_2 kt} = 0 \quad \forall i_2 \in J,\ \forall k : (pad_i < q_k) || (pad_i > q_k + 1) \tag{2}$$

$$\sum_{i_2 \in J} \sum_{t} Y_{0 i_2 kt} = 1 \quad \forall k \tag{3}$$

$$\sum_{i_1 \in J} \sum_{t} Y_{i_1 0 kt} = 1 \quad \forall k \tag{4}$$

$$\sum_{i_2 \in J^0} Y_{i_2 i_1 kt} - \sum_{i_2 \in J^0} Y_{i_1, i_2, k, (t + p_i + s_{i_1 i_2})} = 0 \quad \forall i_1 \in J, \forall k, \forall t \tag{5}$$

$$Y_{i_1 i_2 kt} \in \{0, 1\} \quad i_1, i_2 \in J^0, \forall k \in C, \forall t \in T : i_1 \neq i_2 \tag{6}$$

The objective is to minimize total weighted completion time of reclaiming stockpiles. The processing order of tasks on a reclaimer can be viewed as a route of a vehicle from the perspective of vehicle routing problem. By constraint (1), we ensure that each node is visited once. In other words, each task must be processed on one reclaimer by starting at one time point. Constraint (2) prevents the ineligible stockpile to reclaimer assignments. Constraints (3) and (4) state that the route for each vehicle starts and ends its route at dummy task. Flow balance constraint (5) ensures that if a vehicle visits a node, then it must leave that node as well, excluding the dummy task. Lastly, constraint (6) defines the domain of $Y_{i_1 i_2 kt}$ variables. In this formulation, sub-tours are prevented by flow balance constraint (5) which also keeps account of start times, differently from the network flow formulation of scheduling problems which require additional sub-tour elimination constraints [27].

Proposition 2: The proposed formulation prevents sub-tours.

Proof: Assume that there is a sub-tour of $i_1 - i_2 - i_1$ for a vehicle (i.e., reclaimer) in a feasible solution to the formulation, where i_1 and i_2 are not dummy tasks, and their processing times are non-zero. In that case, if i_1 starts at time t_1, then t_2 starts at $t + s_{i_1} + s_{i_1 i_2}$ by flow balance constraint (5). In that sub-tour, i_1 is immediately visited after i_2. Then, i_1 starts at time $t_1 + s_{i_1} + s_{i_1 i_2} + p_{i_2} + s_{i_2 i_1}$. However, $p_{i_2} + s_{i_2 i_1} > 0$. It contradicts the initial assumption

since a task can only start at one and only one time point by constraint (1). □

2) LOWER BOUND FOR RSP WITH STACKING OPERATION

$[RSP-LB]$ is modified such that the set of tasks (J) with J' as well as parameters p_i and w_i with p'_i and w'_i, respectively. Furthermore, we introduce parameter δ and set of precedences γ. In this variant of the problem, a stockpile must be stacked in the stockyard before its reclaiming starts. Therefore, we add the following precedence constraint into the formulation.

$$\sum_{i_3}\sum_{k}\sum_{t} tY_{i_3 i_1 kt} - \sum_{i_3}\sum_{k}\sum_{t} tY_{i_3 i_2 kt} \geq p_{i_1} \quad \forall (i_1, i_2) \in \gamma \qquad (7)$$

Note that Preposition 1 does not hold for this case because of the precedence relationships among tasks assigned to different machines. Therefore, we extend the variable space with $Y_{i_1 i_1 kt}$ to represent the idleness of machine k at time t after processing task i_1. For modeling purposes, we set $s_{i_1 i_1} = 1 - p_i$, thus flow balance constraint (5) is still satisfied when a machine is idle. In addition, we add the following valid inequality that simply removes the variables in which reclaiming task of a stockpile is an immediate predecessor of its stacking task.

$$\sum_{k}\sum_{t} Y_{i_2 i_1 kt} = 0 \quad \forall (i_1, i_2) \in \gamma \qquad (8)$$

C. UPPER BOUND FOR RSP: CONSTRAINT PROGRAMMING FORMULATION

Constraint programming (CP) is shown to be a proper tool for dealing with scheduling problems with non-crossing constraint [7]. That is why we develop a CP formulation to generate good feasible solutions for RSP.

CP provides a framework for solving solve combinatorial satisfaction and optimization problems. In general, the method can be viewed as a combination of defining constraints over problem variables, finding a set of values from the domains of variables that satisfy constraints, and searching for good solutions via backtracking algorithms. In CP, constraints are actively used to infer new constraints and to reduce the domains of variables. This is performed by a technique called constraint filtering: when domain of a variable is modified, constraints interacting with this variable are investigated to modify domains of other variables to remove inconsistencies.

Furthermore, CP offers rich modeling concepts for compact representation and effective solutions of combinatorial problems. In the proposed formulation, we utilize some of these concepts; namely, interval variables, sequence variables, and global constraints.

Definition 2: **Interval variable** specifies an interval of time during which a task is executed. Each interval variable is characterized by a start time, an end time, a processing time, and a presence status.

Definition 3: **Sequence variable** is defined to represent ordering of a specific set interval variables.

Definition 4: **Global constraint** can represent complex relationships between the problem variables as a single constraint. Global constraints deal with many problem variables at the same time and provide an effective domain reduction by using specialized filtering algorithms.

For more information on constraint programming technique, we kindly refer readers to [28] and [29]. In the following, we introduce a novel CP formulation for RSP and associated decision variables.

Θ_i An interval variable that represents the reclaiming of stockpile i. Its domain is $[1, |T|]$

Ω_{ik} An optional interval variable that represents the reclaiming of stockpile i by reclaimer k. This variable is present in the model if and only if reclaimer k is assigned to stockpile i. Otherwise, its domain will be $\emptyset$.

$\Psi_{i_1 i_2 k_1 k_2}$ A sequence variable that is used for non-crossing constraint, and it determines relative order of interval variables $\Omega_{i_1 k_1}$ and $\Omega_{i_2 k_2}$. Its domain consists of the permutation of the order of these interval variables.

1) UPPER BOUND FOR RSP WITHOUT STACKING OPERATIONS

$$[RSP-UB]:$$

$$\text{minimize} \sum_{i_1 \in J} w_i End(\Theta_i)$$

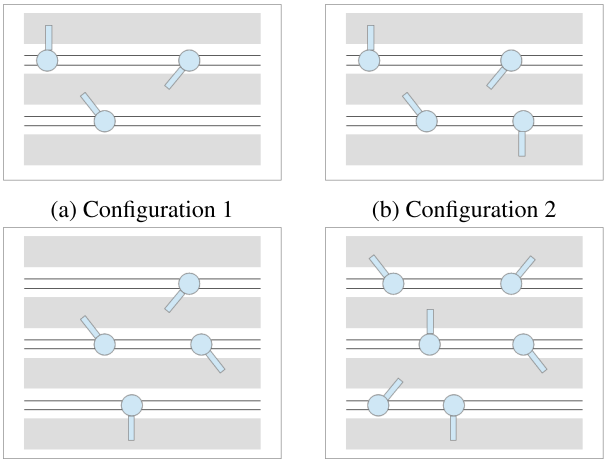
subject to:

$$Alternative(\Theta_i, (\Omega_{ik} | \forall k : (pad_i = q_k) || (pad_i = q_k + 1))) \quad \forall i \qquad (9)$$

$$Disjunctive((\Omega_{ik} | \forall i), s_{i_1 i_2}) \quad \forall k \qquad (10)$$

$$Disjunctive(\Psi_{i_1 i_2 k_1 k_2}) \quad \forall i_1, i_2, \forall k_1, k_2 : q_{k_1} = q_{k_2}, k_{i_1} < k_{i_2}, \quad pad_{i_1} - pad_{i_2} \in \{-1, 0, 1\}, pos_{i_1} + k_{i_1} > pos_{i_2} \qquad (11)$$

Global constraint *Alternative* (8) assigns each stockpile to one of the eligible reclaimers and links the domains of related Θ_i with Ω_{ik} variables. By this constraint, the domain of Ω_{ik} variable is updated as an empty set if reclaimer k is not assigned to stockpile i. Global constraint *Disjunctive* (9) ensures non-overlapping of stockpiles that are assigned to same machine while considering travel times. We imply non-crossing constraint by (10). This constraint also respects the spatial restriction caused by reclaimers performing long travel bench cuts.



(a) Configuration 1 (b) Configuration 2

(c) Configuration 3 (d) Configuration 4

FIGURE 3. Illustration of different stockyard configurations.

2) UPPER BOUND FOR RSP WITH STACKING OPERATIONS

We modify sets and parameters as in Section III-B2 and add the following precedence constraint to $[RSP - UB]$.

$$EndBeforeStart(\Omega_{i_1k_1}, \Omega_{i_2k_2}) \quad \forall (i_1, i_2) \in \gamma,\ \forall k_1,\ k_2 \qquad (12)$$

IV. COMPUTATIONAL EXPERIMENTS

We design a set of computational experiments for assessing the performance of the proposed methods by generating following stockyard configurations with varying numbers of stocking pads ($|A|$), rail tracks ($|W|$), and reclaimers ($|C|$). These configurations are depicted in Figure 3.

- **Configuration 1:** $|A| = 3$, $|W| = 2$, $|C| = 3$. Two reclaimers on rail track 1 and a single reclaimer on rail track 2.
- **Configuration 2:** $|A| = 3$, $|W| = 2$, $|C| = 4$. Two reclaimers on each rail track.
- **Configuration 3:** $|A| = 4$, $|W| = 3$, $|C| = 4$. Two reclaimers on rail track 2 and a single reclaimer on rail tracks 1 and 3.
- **Configuration 4:** $|A| = 4$, $|W| = 3$, $|C| = 6$. Two reclaimers on each rail track.

We assume a discretized stocking pad consisting of $|U| = 60$ 33.33 meters-long stocking positions. That is, the total length of a pad is 2000 meters. This discretization does not change the complexity of proposed formulations but eases the instance generation process. Then, we generate stockpiles such that their lengths (k_i) are uniformly distributed between 2 and 10 units of stocking positions, and reclaiming times (p_i) are proportional to their length, i.e. $p_i = 6k_i$. To generate a feasible allocation of stockpiles, we generate stockpiles one by one by starting from the leftmost of the first pad until stockyard is full. We generate 10 instances for each configuration, and the average number of tasks per instance is 28.8 for configurations 1 and 2, and is 40 for configurations 3 and 4.

In reality, travel times are small compared to reclaiming times. For this reason, we calculate the travel time between stockpiles i_1 and i_2 as their distance on x-axis from midpoints divided by 10 (in unit times). As a result, travel times are bounded within 1 and 6 unit times.

We generate 10 instances for each configuration and compare results of relaxed MIP formulation $[RSP - LB]$ with those of CP formulation $[RSP - UB]$. As CP method has random components on its search phase, we perform with five different trials for each instance. Recall that we obtain a relaxed MIP formulation for deriving a lower bound by relaxing non-crossing constraint of RSP. Accordingly, we get an equivalent CP formulation to $[RSP - LB]$ by removing non-crossing constraint from $[RSP-UB]$. By this way, we can also compare the performance of CP method without including the deviation caused by non-crossing constraint. We set a time limit of a minute and an hour for each CP trial and LB formulation, respectively. All experiments are performed with IBM CPLEX 12.8. Note that all data files are available at IEEE DataPort for reproduction of experiments.

A. RESULTS FOR RSP WITHOUT STACKING OPERATIONS

Table 1 presents average results of computational experiments for RSP without stacking operations. In this table (and throughout the paper), solution times are given in seconds. The individual results for each instance can be found in Appendix.

Lower bound formulation solves all instances to the optimality within time limit of an hour. Average solution times for configurations 3-4 are distinctly higher than those of configurations 1-2. This is not unexpected because our instance generation scheme suggests the number of stockpiles in an instance mainly depends on number of stocking pads, as in reality. For the instances with same number of stocking pads, solution times increase with number of reclaimers. The solution times reported Table 1 are yet considerably short given the complexity of the scheduling problem.

The good performance of the lower bound formulation is due to the very strong linear programming (LP) relaxation of the proposed arc-time-indexed formulation: the average deviation of LP relaxations from the optimal are approximately 0.04%, and LP relaxation yields optimal solution in more than 80% of all instances.

We present results for the computational experiments of CP method for RSP in $[RSP - UB]$ column, in terms of the percentage deviation from the corresponding lower bound values. Average deviation from the lower bound is observed as 0.78%, and it increases for the configurations with larger number of reclaimers per track as there are more interference. In all 40 instances, the worst deviation value is less than just 1.7%. Moreover, it is robust: the maximum relative standard deviation between trials is observed as 0.45%.

In column $[RSP - UB']$, we also present the results for CP formulation without non-crossing constraint, which is equivalent to lower bound formulation $[RSP - LB]$. We observe 0.43% deviation from the lower bound on

TABLE 1. Results for RSP without stacking operations.

Configuration	$[RSP\text{-}LB]$		$[RSP\text{-}UB']$				$[RSP\text{-}UB]$			
	$\%_{LP}$	time	$\%_{LB}^{min.}$	$\%_{LB}^{avg.}$	$\%_{LB}^{max.}$	$\%_{RSD}$	$\%_{LB}^{min.}$	$\%_{LB}^{avg.}$	$\%_{LB}^{max.}$	$\%_{RSD}$
1	0.02	220.2	0.03	0.16	0.27	0.16	0.13	0.45	0.85	0.33
2	0.03	284.1	0.03	0.21	0.38	0.51	0.42	0.85	1.20	0.51
3	0.07	517.6	0.12	0.11	0.17	0.29	0.15	0.59	1.11	0.39
4	0.05	918.7	0.20	0.44	0.65	0.24	0.53	1.21	1.67	0.58
avg.	0.04	485.2	0.10	0.23	0.37	0.30	0.31	0.78	0.97	0.45

TABLE 2. Comparison of different formulations.

Instance		$[RSP\text{-}LB]$	$[RSP\text{-}LB']$		$[GM2012]$		
	z^*	time	time	$\%_{time}$	LB	UB	$\%_{opt}$
1	16930	98	189	-48.1	9145.3	17363	47.3
2	14300	31	127	-75.6	8757.2	14550	39.8
3	12916	51	125	-59.2	7470.7	12976	42.4
4	14269	62	104	-40.4	7675.2	14604	47.4
5	13903	50	107	-53.3	8303.0	14168	41.4
6	15320	1107	1416	-21.8	8790.1	15550	43.3
7	11887	121	147	-17.7	7513.9	12026	37.5
8	14269	109	184	-40.8	8759.3	14451	39.4
9	15504	191	205	-6.8	8958.9	15793	43.3
10	15653	134	163	-17.8	8738.0	15866	44.9
avg.	14495.1	195.4	276.7	-38.1	8362.2	14602.3	42.7

TABLE 3. Results for RSP with stacking operations.

Configuration	$[RSP\text{-}LB]$		$[RSP\text{-}UB']$				$[RSP\text{-}UB]$			
	$\%_{LP}$	time	$\%_{LB}^{min.}$	$\%_{LB}^{avg.}$	$\%_{LB}^{max.}$	$\%_{RSD}$	$\%_{LB}^{min.}$	$\%_{LB}^{avg.}$	$\%_{LB}^{max.}$	$\%_{RSD}$
1	0.00	103.8	0.00	0.08	0.32	0.08	0.10	0.68	1.98	0.19
2	0.02	427.5	0.00	0.15	0.47	0.13	0.25	1.43	2.93	0.60
3	0.05	461.1	0.07	0.21	0.54	0.22	0.23	0.96	1.65	0.23
4	0.13	1561.9	0.00	0.48	0.91	0.27	1.00	2.52	4.75	0.61
avg.	0.05	638.6	0.02	0.23	0.56	0.18	0.40	1.39	2.82	0.41

average, by excluding the effect of non-crossing constraint on the performance measure. This indicates that proposed CP formulation for RSP is able to cope with the challenging non-crossing constraint effectively.

Furthermore, we compare $[RSP - LB]$ and the formulation with the variables that address idleness of reclaimers $[RSP - LB']$ to measure the impact of Proposition 1 on the solution performance. Table 2 presents the results for these formulations as well as for the MIP formulation presented in Gokhale and Mathirajan [21] -$[GM2012]$- by using the set of instances with Configuration 2.

Results show that we are able to speed up the solution process by 38.1% with Proposition 1. In addition to that, $[RSP - LB]$ strongly outperform the only exact approach appears in the literature for this scheduling problem. That is, $[RSP-LB]$ is able to solve all instances while $[GM2012]$ fails to solve any of the instances within a time limit. As $[GM2012]$ cannot obtain optimal result, we only report lower and upper bound values at the termination of solution process. While this formulation is able to find incumbent values as near as 1.59% to optimal values on average, we observe large optimality gaps due to its weak LP relaxation.

B. RESULTS FOR RSP WITH STACKING OPERATIONS

The set of instances generated in previous section can be straightforwardly modified for this case. However, resulting instances will be significantly larger because (i) there is also a stacking task for each stockpile and (ii) the length of schedule $|T|$ increases. In addition, precedence constraint adds large numbers of constraints and non-zero coefficients to the formulation. For this reason, we perform experiments with a set of smaller instances. Specifically, we assume that the total length of each pad to be 1000 meters instead of 2000 to reduce the number of stockpiles by half.

TABLE 4. Detailed results for RSP without stacking operation.

		[RSP-LB]				[RSP-UB]		
	ins	z^*_{LP}	z^*	$\%_{LP}$	time	z_{avg}	$\%_{LB}$	$\%_{RDS}$
Configuration 1	1	21612.0	21630	0.08	668	21705.8	0.35	0.20
	2	18173.0	18173	0.00	310	18310.5	0.75	0.06
	3	16820.0	16820	0.00	52	16842.3	0.13	0.23
	4	18442.0	18442	0.00	65	18537.0	0.51	0.41
	5	17725.0	17725	0.00	69	17850.6	0.71	0.20
	6	19558.0	19558	0.00	54	19622.5	0.33	0.59
	7	15124.0	15124	0.00	57	15188.9	0.43	0.90
	8	18104.8	18112	0.04	142	18233.8	0.67	0.25
	9	19710.3	19724	0.07	595	19802.0	0.39	0.15
	10	20150.0	20150	0.00	190	20189.8	0.19	0.26
	avg			0.02	220.2	18628.3	0.45	0.33
Configuration 2	1	16930.0	16930	0.00	98	17001.5	0.42	0.61
	2	14300.0	14300	0.00	35	14392.6	0.64	0.44
	3	12916.0	12916	0.00	51	13028.6	0.87	0.67
	4	14269.0	14269	0.00	62	14435.0	1.16	0.56
	5	13902.0	13902	0.00	50	14068.4	1.20	0.50
	6	15271.3	15320	0.32	1907	15406.3	0.56	0.32
	7	11887.0	11887	0.00	165	12008.7	1.02	0.47
	8	14269.0	14269	0.00	124	14377.7	0.76	0.16
	9	15504.0	15504	0.00	205	15657.5	0.99	0.45
	10	15653.0	15653	0.00	144	15786.1	0.85	0.86
	avg.			0.03	284.1		0.85	0.51
Configuration 3	1	24842	24842	0.00	87	24879.1	0.15	0.19
	2	23190.0	23190	0.00	77	23448.1	1.11	0.44
	3	23862.0	23862	0.00	135	23918.6	0.23	0.38
	4	35469.8	35505	0.10	1055	35663.7	0.45	0.53
	5	25361.1	25376	0.06	1930	25581.3	0.81	0.69
	6	30612.0	30612	0.00	157	30830.5	0.71	0.40
	7	33768.0	33768	0.00	173	34120.2	1.04	0.42
	8	27988.0	28105	0.42	1148	28281.0	0.62	0.58
	9	34061.0	34061	0.00	100	34207.5	0.43	0.11
	10	31081.2	31851	0.16	1314	31963.8	0.35	0.10
	avg.			0.07	617.6		0.59	0.39
Configuration 4	1	17668.0	17668	0.00	360	17763.3	0.54	0.33
	2	16426.0	16426	0.00	215	16646.3	1.34	0.45
	3	16938.0	16938	0.00	366	17089.8	0.89	0.29
	4	24872.1	24891	0.08	1295	25200.0	1.24	1.20
	5	17970.0	17982	0.07	1454	18157.1	0.97	1.04
	6	21826.1	21851	0.11	486	22127.6	1.27	0.39
	7	23978.0	23978	0.00	452	24333.4	1.48	0.33
	8	19658.3	19673	0.08	1790	19982.3	1.57	0.79
	9	23435.0	23435	0.00	95	23698.5	1.12	0.42
	10	23178.5	23208	0.13	2674	23595.0	1.67	0.50
	avg.			0.05	918.7		1.21	0.58

TABLE 5. Detailed results for RSP with stacking operation.

		[RSP-LB]				[RSP-UB]		
	ins	z^*_{LP}	z^*	$\%_{LP}$	time	z_{avg}	$\%_{LB}$	$\%_{RDS}$
Configuration 1	1	8001.0	8001	0.00	65	8009.0	0.10	0.22
	2	5871.0	5871	0.00	76	5893.2	0.38	0.16
	3	5970.0	5970	0.00	120	5993.0	0.39	0.37
	4	6901.0	6901	0.00	136	6929.3	0.41	0.44
	5	7530.0	7530	0.00	80	7679.6	1.98	0.09
	6	6795.0	6795	0.00	77	6830.0	0.52	0.00
	7	5705.0	5705	0.00	160	5812.8	1.89	0.05
	8	7450.0	7450	0.00	72	7487.7	0.51	0.07
	9	7251.0	7251	0.00	95	7262.0	0.15	0.06
	10	7332.0	7332	0.00	157	7366.0	0.46	0.45
	avg			0.00	103.8		0.68	0.19
Configuration 2	1	6298.0	6298	0.00	64	6351.3	0.84	0.30
	2	5650.0	5650	0.00	99	5715.4	1.16	0.31
	3	3930.0	3930	0.00	285	3990.5	1.53	0.52
	4	5511.2	5513	0.03	1100	5582.9	1.26	0.98
	5	5843.0	5843	0.00	1168	6014.7	2.94	0.43
	6	5484.0	5484	0.00	156	5560.6	1.40	1.00
	7	4563.0	4563	0.00	359	4662.5	2.18	1.11
	8	5455.0	5465	0.18	725	5479.0	0.25	0.00
	9	6012.0	6012	0.00	93	6117.8	1.76	0.43
	10	5939.0	5939	0.00	126	5998.9	1.01	0.90
	avg.			0.02	427.5		1.43	0.60
Configuration 3	1	7468.7	7504	0.47	1338	7558.4	0.23	0.37
	2	7541.0	7541	0.00	108	7558.4	0.23	0.37
	3	7300.1	7305	0.07	929	7416.0	1.52	0.13
	4	13978.0	13978	0.00	175	14094.3	0.83	0.51
	5	7245.0	7245	0.00	725	7368.6	1.71	0.40
	6	10122.1	10125	0.03	557	10223.7	0.97	0.05
	7	9042.0	9042	0.00	173	9191.7	1.66	0.17
	8	9770.0	9770	0.00	182	9818.1	0.49	0.21
	9	11728.0	11728	0.00	124	11823.0	0.81	0.00
	10	13417.0	13417	0.00	300	13465.6	0.36	0.45
	avg.			0.06	461.1		0.96	0.23
Configuration 4	1	6003.8	6045	0.68	1936	6181.8	2.26	0.21
	2	4972.0	4972	0.00	576	5208.7	4.75	0.23
	3	6491.4	6495	0.06	1542	6670.6	2.70	1.06
	4	9165.2	9171	0.06	1533	9405.8	2.56	0.89
	5	6273.5	6282	0.14	917	6479.8	3.14	0.22
	6	8065.0	8077	0.15	3051	8303.3	2.80	0.45
	7	6212.7	6214	0.02	960	6325.6	1.79	0.77
	8	7052.6	7059	2118	0.09	7200.1	2.00	0.99
	9	7553.0	7553	0.00	331	7628.8	1.00	0.46
	10	8322.9	8333	0.12	2655	8517.4	2.21	0.71
	avg.			0.13	1561.9		2.52	0.60

We present the results in Table 3. It shows that the CP formulation can effectively solve this extension of the problem with a similar performance to the results presented in Table 1. The only distinct difference is observed in the deviation of average objective values of $[RSP-UB]$ from the lower bound. The elevation in these values can be attributed having the

same number of tasks (stacking + reclaiming) in a shorter pad. Thus, there are more interference among reclaimers mounted on a same rail track. On the other hand, the CP formulation without non-crossing constraint still finds near optimal values as in the case of RSP without stacking operations. This suggests that optimality gaps for $[RSP-UB]$ are possibly much lower than the reported deviation from the lower bound.

Furthermore, we also perform experiments with the following disaggregated precedence constraint presented by Christofides *et al.* [30], instead of (7).

$$\sum_{i_3}\sum_{k}\sum_{s\in\{1,\ldots,t-p_{i_1}\}} Y_{i_3 i_1 k s} - \sum_{i_3}\sum_{k}\sum_{s\in\{1,\ldots,t\}} Y_{i_3 i_2 k s} \geq 0 \quad \forall (i_1, i_2) \in \gamma,\ \forall t \qquad (13)$$

Artigues [31] notes that time-indexed formulations with constraint (12) has a stronger LP relaxation than those with constraint (7). At the same time, however, this constraint is computationally expensive as it checks the precedence relationships among tasks for each discrete time period. Besides, arc-time indexed formulations are even larger in size than pure time-indexed formulations. In our experiments with constraint (12), all instances are either terminated with memory limitations or its LP relaxation could not be calculated within time limit. By this way, we show the impracticality of using disaggregated precedence constraints with arc-time indexed formulations.

V. CONCLUSION

This paper presents an effective lower and upper bound for reclaimer scheduling problem which is a variant of PMSP with non-crossing constraint, sequence dependent setup times, eligibility restrictions, and precedence relationships.

As a lower bound, we develop an MIP formulation for exact solution of PMSP with sequence dependent setup times and eligibility restrictions after observing a dominance property that reduces the size search space by eliminating solutions with unforced idleness of machines. The performance of this formulation also demonstrates the appropriateness of arc-time-indexed formulation for modeling and solving complex scheduling problems involving sequence dependent setup times. It should be noted that this formulation goes beyond providing a lower bound to RSP, as it effectively solves a scheduling problem with a practical relevance.

The performance of LB formulation for RSP with stacking operations are limited compared to the formulation without stacking operations because of the vast increase in problem size. Therefore, future research should be performed to solve this formulation more effectively. One natural approach would be to decompose the problem such that precedence constraints are handled separately.

As an upper bound, we present a robust constraint programming formulation that provides near optimal results in a minute. Thus, in addition to generating reclaimer schedules, it can be further utilized by terminal management as a decision support tool for evaluating the impact of different stockyard designs empirically. In some terminals, stacking and reclaiming operations are performed by designated equipment. Accordingly, the applicability of methods presented in this paper should be investigated for such a setting.

REFERENCES

[1] *Review of Maritime Transport*, UNCTAD, Geneva, Switzerland, 2018.
[2] D. Steenken, S. Voß, and R. Stahlbock, "Container terminal operation and operations research—A classification and literature review," *OR Spectr.*, vol. 26, no. 1, pp. 3–49, Jan. 2004.
[3] G. Belov, N. L. Boland, M. W. P. Savelsbergh, and P. J. Stuckney, "Logistics optimization for a coal supply chain," *J Heuristics*, vol. 26, no. 2, pp. 269–300, 2020.
[4] A. Allahverdi, C. T. Ng, T. C. E. Cheng, and M. Y. Kovalyov, "A survey of scheduling problems with setup times or cost," *Eur. J. Oper. Res.*, vol. 187, no. 3, pp. 985–1032, 2008.
[5] J. Y.-T. Leung and C.-L. Li, "Scheduling with processing set restrictions: A survey," *Int. J. Prod. Econ.*, vol. 116, no. 2, pp. 251–262, Dec. 2008.
[6] N. Boysen, D. Briskorn, and F. Meisel, "A generalized classification scheme for crane scheduling with interference," *Eur. J. Oper. Res.*, vol. 258, no. 1, pp. 343–357, Apr. 2017.
[7] O. Unsal and C. Oguz, "Constraint programming approach to quay crane scheduling problem," *Transp. Res. E, Logistics Transp. Rev.*, vol. 59, pp. 108–122, Nov. 2013.
[8] Y. Zhu and A. Lim, "Crane scheduling with non-crossing constraint," *J. Oper. Res. Soc.*, vol. 57, no. 12, pp. 1464–1471, Dec. 2006.
[9] D. Hu and Z. Yao, "Stacker-reclaimer scheduling in a dry bulk terminal," *Int. J. Comput. Integr. Manuf.*, vol. 25, no. 11, pp. 1047–1058, Nov. 2012.
[10] C. Wang, X.-W. Lu, and R. Sitters, "Scheduling reclaimer operations in the stockyard to minimize makespan," *Acta Mathematicae Applicatae Sinica, English Ser.*, vol. 34, no. 3, pp. 597–609, Jul. 2018.
[11] T. van Vianen, J. Ottjes, and G. Lodewijks, "Simulation-based determination of the required stockyard size for dry bulk terminals," *Simul. Model. Pract. Theory*, vol. 42, pp. 119–128, Mar. 2014.
[12] J. Xin, R. R. Negenborn, and T. van Vianen, "A hybrid dynamical approach for allocating materials in a dry bulk terminal," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 3, pp. 1326–1336, Jul. 2018.
[13] E. Angelelli, T. Kalinowski, R. Kapoor, and M. W. P. Savelsbergh, "A reclaimer scheduling problem arising in coal stockyard management," *J. Scheduling*, vol. 19, no. 5, pp. 563–582, Oct. 2016.
[14] T. Kalinowski, R. Kapoor, and M. W. P. Savelsbergh, "Scheduling reclaimers serving a stock pad at a coal terminal," *J. Scheduling*, vol. 20, no. 1, pp. 85–101, Feb. 2017.
[15] L. Tang, D. Sun, and J. Liu, "Integrated storage space allocation and ship scheduling problem in bulk cargo terminals," *IIE Tran.*, vol. 48, no. 5, pp. 428–439, 2016.
[16] G. C. Menezes, G. R. Mateus, and M. G. Ravetti, "A branch and price algorithm to solve the integrated production planning and scheduling in bulk ports," *Eur. J. Oper. Res.*, vol. 258, no. 3, pp. 926–937, May 2017.
[17] R. L. Burdett, P. Corry, C. Eustace, and S. Smith, "A flexible job shop scheduling approach with operators for coal export terminals—A mature approach," *Comput. Oper. Res.*, vol. 115, Mar. 2020, Art. no. 104834.
[18] O. Unsal and C. Oguz, "An exact algorithm for integrated planning of operations in dry bulk terminals," *Transp. Res. E, Logistics Transp. Rev.*, vol. 126, pp. 103–121, Jun. 2019.
[19] C. Bierwirth and F. Meisel, "A fast heuristic for quay crane scheduling with interference constraints," *J. Scheduling*, vol. 12, no. 4, pp. 345–360, Aug. 2009.
[20] J. H. Chen, D.-H. Lee, and M. Goh, "An effective mathematical formulation for the unidirectional cluster-based quay crane scheduling problem," *Eur. J. Oper. Res.*, vol. 232, no. 1, pp. 198–208, Jan. 2014.
[21] R. Gokhale and M. Mathirajan, "Scheduling identical parallel machines with machine eligibility restrictions to minimize total weighted flowtime in automobile gear manufacturing," *Int. J. Adv. Manuf. Technol.*, vol. 60, nos. 9–12, pp. 1099–1110, Jun. 2012.
[22] Y. Unlu and S. J. Mason, "Evaluation of mixed integer programming formulations for non-preemptive parallel machine scheduling problems," *Comput. Ind. Eng.*, vol. 58, no. 4, pp. 785–800, May 2010.

[23] K.-C. Ying, C.-Y. Cheng, S.-W. Lin, and C.-Y. Hung, "Comparative analysis of mixed integer programming formulations for single-machine and parallel-machine scheduling problems," *IEEE Access*, vol. 7, pp. 152998–153011, 2019.
[24] F. Sourd, "New exact algorithms for one-machine earliness-tardiness scheduling," *INFORMS J. Comput.*, vol. 21, no. 1, pp. 167–175, Feb. 2009.
[25] A. Pessoa, E. Uchoa, M. P. de Aragao, and R. Freitas, "Exact algorithm over an arc-time indexed formulations for parallel machine scheduling problems," *Math. Program. Comput.*, vol. 2, pp. 259–290, 2010.
[26] I. M. Ovacik and R. Uzhoy, "Worst-case error bounds for parallel machine scheduling problems with bounded sequence-dependent setup times," *Oper. Res. Lett.*, vol. 14, no. 5, pp. 251–256, Dec. 1993.
[27] B. Cesaret, C. Oguz, and F. S. Salman, "A tabu search algorithm for order acceptance and scheduling problem," *Comput. Oper. Res.*, vol. 39, no. 6, pp. 1197–1205, 2012.
[28] K. R. Apt, *Principles of Constraint Programming*. London, U.K.: Cambridge Univ. Press, 2003.
[29] P. Laborie, J. Rogerie, P. Shaw, and P. Vilim, "IBM ILOG CP optimizer for scheduling," *Constraints*, vol. 23, no. 2, pp. 1–41, 2018.
[30] N. Christofides, R. Alvarez-Valdes, and J. M. Tamarit, "Project scheduling with resource constraints: A branch and bound approach," *Eur. J. Oper. Res.*, vol. 29, no. 3, pp. 262–273, Jun. 1987.
[31] C. Artigues, "On the strength of time-indexed formulations for the resource-constrained project scheduling problem," *Oper. Res. Lett.*, vol. 45, no. 2, pp. 154–159, 2017.
[32] O. Unsal, "Mathematical models for maritime terminal operations," Ph.D. dissertation, GSSE, Koç Univ., stanbul, Turkey, 2019.

ÖZGÜR ÜNSAL was born in Aydin, Turkey, in 1988. He received the B.S. degree in industrial system engineering from the University of Economics, Izmir, Turkey, in 2010, and the M.S. and Ph.D. degrees in industrial engineering and operations management from Koç University, Istanbul, Turkey, in 2013 and 2019, respectively.

His research interests include the development of hybrid optimization algorithms for complex scheduling problems and improving the access to healthcare.