Optimising Reclaimer Schedules

M. Savelsbergh [a], **R. Kapoor** [a]

[a] *School of Mathematical and Physical Sciences, University of Newcastle, Australia*
Email: reena.kapoor@uon.edu.au

Abstract: The Hunter Valley Coal Chain (HVCC), with 40 coal mines, 30 load points, 3 coal loading terminals (9 ship berths and 7 loaders), and loading about 1,500 ships annually, is one of the largest coal export operations in the world. All planning and scheduling of coal exports in the Hunter Valley is managed by the Hunter Valley Coal Chain Coordinator Limited (HVCCC) with a main goal of maximizing the throughput. Effective management of the stockyards at the coal terminals is critical to achieving this goal. Coal arrives at a coal terminal by train. The coal is dumped and stacked to form stockpiles. Coal brands are a blended product, with coal from different mines having different characteristics "mixed" in a stockpile to meet the specifications of the customer. Once the ship for which the coal is destined arrives at a berth at the terminal, the coal is reclaimed and loaded onto the ship. The ship then transports the coal to its destination. The efficiency of a stockyard depends heavily on the reclaimers' productivity. Thus, effective scheduling of the reclaimers is of crucial importance.

To better understand the fundamental difficulties of reclaimer scheduling, we are investigating a number of variants of an abstract reclaimer scheduling problem. In this paper, we consider a variant in which two reclaimers serve two parallel identical stock pads, i.e., the reclaimers move back and forth along the length of the pads and reclaim stockpiles from both pads. However, since the two reclaimers move on the same rails they cannot pass each other. The goal is to reclaim a set of stockpiles with given positions on the pads as quickly as possible.

More specifically, one reclaimer starts at one end of the stock pads and the other reclaimer starts at the other end of the stock pads. After reclaiming the stockpiles, the reclaimers need to return to their original position. The reclaimers are identical and thus have the same reclaim speed and the same travel speed. The reclaimers cannot pass each other. A set of stockpiles positioned on the two stock pads has to be reclaimed. Each stockpile has a start and end position and thus a length. When a stockpile is reclaimed, it has to be traversed along its entire length by one of the reclaimers, either from left to right or from right to left. The reclaim time of a stockpile is determined by its length and the reclaim speed of the reclaimers. The goal is to reclaim the stockpiles and minimise the maximum of the return time of two reclaimers to their original positions.

In Angelelli et al. (2013), we have shown that this variant of the reclaimer scheduling problem is NP-complete and have introduced three approximation algorithms for its solution. The three approximation algorithms use different rules for deciding which stockpiles are to be served by each of the reclaimers, but in all three algorithms the reclaimers employ a simple out-and-back routing strategy to reclaim their assigned stockpiles. To decide on the assignment of stockpiles to reclaimers, the three algorithms divide each of the pads into two parts and assign the stockpiles on left of the dividing point to the left reclaimer and the remaining stockpiles to right reclaimer. The algorithms differ in how the dividing points are chosen.

In this paper, we discuss the results of a computational study in which the performance of implementations of the different approximation algorithms are compared on randomly generated instances. The results demonstrate that high-quality solutions can be obtained efficiently for instances with widely varying characteristics.

Keywords: *Stockyard management, scheduling, routing, approximation algorithm*

1 INTRODUCTION

We investigate a reclaimer scheduling problem that arises in the management of a stockyard of a coal export terminal. Coal is a blended product and is assembled or built at a coal terminal in the form of stockpiles before being loaded on to a ship and transported to their final destination. Stackers are used to add arriving coal to stockpiles in the yard at the terminal, and reclaimers are used to reclaim finished stockpiles so they can be loaded onto ships waiting at the berths. The scheduling of the reclaimers is a critical component in the effective management of a stockyard. We consider a variant in which two reclaimers serve two parallel stock pads, i.e., the reclaimers move back and forth along the length of the pads and reclaim stockpiles from both pads. However, since the two reclaimers move on the same rails they cannot pass each other. The goal is to reclaim a set of stockpiles with given positions on the pads as quickly as possible.

The scheduling of yard cranes and quay cranes in container terminals has some similarities to the scheduling of bucket wheel reclaimers in coal terminals. In both situations, the machines move along a single rail track and thus cannot pass each other, and the machines can only handle object (a container in the case of yard and quay cranes and a stockpile in the case of a bucket wheel reclaimer) at a time. Below, we discuss some of the literature on the scheduling of equipment in container terminals.

Peterkofsky and Daganzo (1990) develop a branch-and-bound algorithm for the static quay crane scheduling problem. Kim and Park (2004) address a quay crane scheduling problem for a single container ship. They propose a branch-and-bound algorithm as well as a greedy randomized adaptive search procedure. Moccia et al. (2006) present of reformulation of the problem studied by Kim and Park (2004) and develop a branch-and-cut algorithm to solve it exactly. The quay crane scheduling problems considered in the papers above do not include a *no-passing* constraint. The first papers to consider quay crane scheduling problems with a no-passing constraint are Ng and Mak (2006) for a single container ship and Liu et al. (2006) for multiple container ships. Liu et al. (2006) focus on minimising the time to unload and load a container ship and they propose a heuristic for solving this problem. Lim et al. (2004) introduce an alternative mathematical model, a branch-and-bound algorithm, and a simulated annealing algorithm for the problem considered by Ng and Mak (2006).

For yard crane scheduling, Young Kim and Hwan Kim (1999) focus on minimising the sum of the set up travel time of cranes in a container storage block by modeling the process as a mixed integer programming model. For the same setting, Kim and Kim (2003) develop heuristic algorithms. In a similar vain, and for the same setting, Linn et al. (2003) and Zhang et al. (2002) present mixed integer programming models and heuristic algorithms for yard crane deployment. Ng and Mak (2005b,a) study the problem of scheduling a yard crane that has to carry out a number of handling jobs with different ready times. The objective is to minimize the sum of job waiting times. Ng (2005) investigates the problem of scheduling multiple yard cranes so as minimise the total ship loading time or the sum of truck waiting times in a yard zone. Petering (2009) develops a simulation study to show that restrictive yard crane movements yield a higher quay crane work rate than a system that allows greater yard crane mobility. Researchers also studied vehicle scheduling for container transportation in a terminal. For a review of this material, we refer the reader to Hu and Yao (2012).

The key differences between reclaimer scheduling and quay/yard crane scheduling are: (1) Bucket wheel reclaimers move along a rail track in between two pads and reclaim stockpiles on either pad in a series of long travel bench cuts where the reclaimer is required to turn around at the end of each cut. On the other hand, quay cranes run on a single rail track and load/unload containers from a vessel on the quay side and from trucks on the yard side by moving horizontally along the rail track, moving vertically to reach the position of container to be served and moving up and down to load/unload a container. Yard cranes are used to load/unload containers from trucks onto a yard block and vice versa using three-dimensional movements similar to quay cranes. (2) For quay and yard cranes the time to move between any two adjacent bays is same whereas the time to move between any two stockpiles can differ for bucket wheel reclaimers.

The only paper that we are aware of that considers the scheduling of equipment in a coal terminal is the recent work by Hu and Yao (2012). They consider the problem of scheduling stackers and reclaimers under the assumptions that between any two adjacent pads there is only one reclaimer to avoid the passing of reclaimers and that the number of stockpiles on each pad is same. We study more general and challenging settings.

The remainder of the paper is organized as follows. In Section 2, we present definitions and notation. In Section 3, we describe the approximation algorithms. Finally, in Section 4, we present the result of our computational study.

2 DEFINITIONS AND NOTATION

We consider the following reclaimer scheduling problem. There are two identical reclaimers R_0 and R_1 that serve two stock pads; one pad on either side of the reclaimers. Reclaimer R_0 starts at one end of the stock pads and reclaimer R_1 starts at the other end of the stock pads. The reclaimers have a reclaim or processing speed p, a travel speed $s \geq p$, cannot pass each other and have to return to their original position. Each stockpile has a given length and a given reclaim time (derived from the stockpile's size and the reclaim speed of the reclaimers). When a stockpile is reclaimed, it has to be traversed along its entire length by one of the reclaimers, either from left to right or from right to left. A set of n stockpiles positioned on the two stock pads, stockpiles $1, \ldots, n_1$ on Pad 1 and stockpiles $n_1 + 1, \ldots, n$ on Pad 2, has to be reclaimed. Each stockpile i has a start position l_i and an end position r_i and thus a length $r_i - l_i$. When a stockpile is reclaimed, it has to be traversed along its entire length by one of the reclaimers, either from left to right or from right to left. The reclaim time of a stockpile i is determined by its length and the reclaim speed of the reclaimers, i.e., $\frac{r_i - l_i}{p}$. The goal is to reclaim the stockpiles and minimize the maximum of the return time of two reclaimers to their original positions.

3 DESCRIPTION OF THE APPROXIMATION ALGORITHMS

Before presenting the approximation algorithms, we observe that the relaxed version of the problem in which preemption is allowed, i.e., in which the reclaiming of a stockpile can be interrupted and resumed later, possibly by a different reclaimer, can be solved easily by dividing the work, i.e., reclaim time and (unavoidable) travel time, equally over the two reclaimers. The value of this preemptive schedule provides a lower bound on the value of the optimal schedule.

There are two types of decisions that have to be made in the reclaimer scheduling problem: assignment decisions, indicating which reclaimer will process a stockpile, and routing decisions, indicating in what sequence a reclaimer will process its assigned stockpiles.

We have designed three approximation algorithms, which employ different rules for deciding which stockpiles are to be served by each of the reclaimers, but in all three algorithms the reclaimers employ a simple out-and-back routing strategy to reclaim their assigned stockpiles. To decide on the assignment of stockpiles to reclaimers, the three algorithms divide each of the pads into two parts and assign the stockpiles on left of the dividing point to the left reclaimer and the remaining stockpiles to right reclaimer. The algorithms differ in how the dividing points are chosen.

In out-and-back routing each reclaimer first processes the assigned stockpiles on one of the pads while moving towards its farthest point and then processes the stockpiles on the other pad while moving back towards its initial position. Note that this gives four possible schedules (because a reclaimer has two choices for the pad to process first and there are two reclaimers). The schedule that leads to the minimum makespan is chosen. (The makespan for the different schedules can differ because of the no-passing constraint, which may force one of the reclaimers to wait.)

SPLIT: This approximation algorithm divides the pads into two sections, a left section and a right section, and assigns the stockpiles in the left section to the left reclaimer R_0 and the stockpiles in the right section to the right reclaimer R_1. In case there is a stockpile which crosses the split point, then the stockpile will be assigned to the left reclaimer R_0 if its major portion is on the left side of the split point, and to the right reclaimer R_1 otherwise. Only the start and end positions of a stockpile are considered as split points. See Algorithm 1 for more details.

In Angelelli et al. (2013) we show that **SPLIT** is a 2-approximation algorithm.

We also consider a slight variation, **SPLIT**$^+$, in which we evaluate two schedules when there is a stockpile i that crosses the split point: assigning stockpile i once to the left reclaimer R_0 and once to the right reclaimer R_1, and take the best of the two schedules.

PARTITION: This approximation algorithm divides *each* pad into two sections, a left section and a right section, and assigns the stockpiles in the left section to the left reclaimer R_0 and the stockpiles in the right section to the right reclaimer R_1. Only the start and end positions of a stockpile are considered as split points. See Algorithm 2 for more details. Note that in **PARTITION** there are no stockpiles that cross a split point.

So far, we have considered only the simplest routing technique, i.e. **Out & Back Routing**, to calculate C_{max} for a given assignment of stockpiles to the two reclaimers. But it is not hard to see that **Out & Back Routing** is not always the best routing strategy. Consider, for example, the instance with four stockpiles of lengths

Algorithm 1 The **SPLIT** algorithm

Input: Start positions of the stockpiles $l_1, ..., l_n$, end positions of stockpiles $r_1, ..., r_n$, processing speed p and traveling speed s. Merge the sequence $l_1, ..., l_n$ and $r_1, ..., r_n$ and sort them in nondecreasing order to obtain the set of potential split points $\alpha_1, ..., \alpha_{2n}$.
Output: Makespan C^*_{max} and S^*

```
Set C*_max = ∞
for i = 1, ..., 2n do
    for k = 1, ..., n do
        if (r_k + l_k)/2 ≤ α_i then
            Assign stockpile i to R_0
        else
            Assign stockpile i to R_1
        end if
    end for
    Calculate C_max for the corresponding schedule S
    if C_max ≤ C*_max then
        Set C*_max = C_max and S* = S
    end if
end for
```

Algorithm 2 The **PARTITION** algorithm

Input:
$J_1 = \{1...n_1\}$, the set of stockpiles on Pad 1, $J_2 = \{n_1+1...n\}$, the set of stockpiles on Pad 2, start positions of the stockpiles $l_1, ..., l_n$, end positions of stockpiles $r_1, ..., r_n$, processing speed p and traveling speed s.
Output: Makespan C^*_{max} and S^*

```
Set C*_max = ∞
for k = 1, ..., n_1 + 1 do
    for h = n_1 + 1, ..., n + 1 do
        Assign to R_0 the subsets J_1^(1) = {j | j < k} and J_2^(1) = {j | n_1 < j < h}
        Assign to R_1 the subsets J_1^(2) = J_1 \ J_1^(1) and J_2^(2) = J_2 \ J_2^(1)
        Calculate C^max for the corresponding schedule S.
        if C_max ≤ C*_max then
            Set C*_max = C_max and S* = S
        end if
    end for
end for
```

$2, 10, 10, 2$ shown in Figure 1. Let the travel speed s be 5 and the processing speed p be 1. Furthermore assume that stockpiles 1 and 3 are assigned to the left reclaimer R_0 and stockpiles 2 and 4 to right reclaimer R_1. **Out & Back Routing** results in $C_{max} = 14.4$. However, when the left reclaimer first travels to r_3

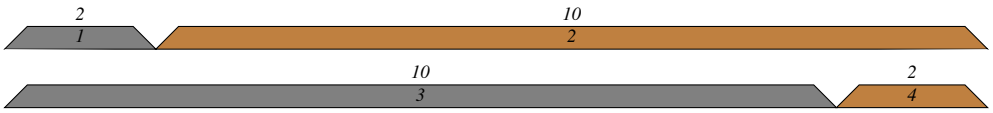

Figure 1. Instance demonstrating that out & back routing is not always optimal.

without processing any stockpile, then processes Stockpile 3 while coming back, then travels to r_1, and finally processes Stockpile 1, and the right reclaimer first processes Stockpile 2, then turns and processes Stockpile 4 on the way back, the resulting C_{max} is 13.6. This shows that sometimes "zigzagging" can be beneficial.

Therefore, we consider two routing improvement strategies. We note that an alternative routing can only result in an improvement if there is waiting in the schedule produced with **Out & Back Routing**. Thus, in the description of the routing improvement strategies, we assume that one of the reclaimers has to wait in schedule S. Without loss of generality, we can assume that its the right reclaimer R_1.

To achieve an improvement, the waiting time needs to be reduced (by increasing the total travel time). Since we have assumed that R_1 is waiting in schedule S, an improvement may be obtained by having R_0 reach its farthest point as early as possible and having R_1 reach its farthest point as late as possible while increasing the total travel time as little as possible. Each of the improvement strategies is designed around this idea.

Smart Out & Back Routing: We start by dividing the pads into the largest number of sections $P_1, P_2, .., P_m$ with the property that none of the stockpiles crosses a section boundary. See Figure 2 for an example. Next, for each section P_i, we calculate the travel and processing time associated with Pad 1 from the start point of the section to the end point of the section, denoted by W_i^1, and, similarly, the travel and processing time associated with Pad 2, denoted by W_i^2. For each section P_i, $i = 1, ..., m$, let $min_i = \min\{W_i^1, W_i^2\}$ and $max_i = \max\{W_i^1, W_i^2\}$. Let the farthest stockpile processed by R_0 on Pad 1 be in section P_k and on Pad 2 be in section P_j. Since the pads are identical, we may assume that $j < k$, i.e., the farthest point reached by R_0 is

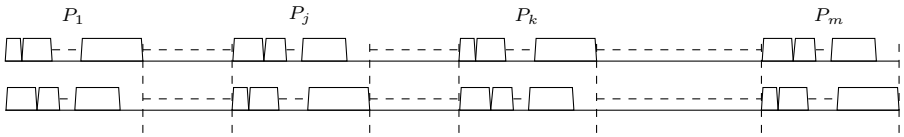

Figure 2. Dividing the pads into sections for **Smart Out & Back Routing**.

determined by a stockpile on Pad 1. Now in each section P_i, $i = 1, ..., k$, R_0 processes its assigned stockpiles among those that determine min_i while going forward and processes its remaining assigned stockpiles while coming back, whereas R_1 does the opposite.

We observe that the total travel time for each reclaimer in the resulting routes has not changed as both reclaimers still process all stockpiles in one forward and one backward pass. However, the waiting time of R_1 may be reduced.

The next improvement technique is a local neighborhood search technique where we try to improve C_{max} by applying local changes in the route corresponding to schedule S.

Restricted ZigZag Routing: We will describe "Restricted ZigZag Routing" for the case in which R_1 incurs waiting time, R_0 processes stockpiles on Pad 1 while going forward and R_1 processes stockpiles on Pad 2 while going forward, because the other cases can be handled in a similar way.

Let us denote the set of stockpiles processed by reclaimer R_0 on Pad 1 going forward by $M_f^0 = \{a_1^0, a_2^0, ..., a_{k_0}^0\}$ and the set of stockpiles processed by reclaimer R_0 on Pad 2 going backwards by $M_b^0 = \{a_{k_0+1}^0, a_{k_0+2}^0, ..., a_{n_0}^0\}$. Similarly let $M_f^1 = \{a_1^1, a_2^1, ..., a_{k_1}^1\}$ denote the set of stockpiles of Pad 2 processed by R_1 going forward and $M_b^1 = \{a_{k_1+1}^1, a_{k_1+2}^1, ..., a_{n_1}^1\}$ denote the set of stockpiles of Pad 1 processed by R_1 going backwards. Note that initially, M_f^0 contains all the jobs on Pad 1 assigned to R_0 and M_b^0 contains all the jobs on Pad 2 assigned to R_0, and M_f^1 contains all the jobs on Pad 2 assigned to R_1 and M_b^1 contains all the jobs on Pad 1 assigned to R_1.

Because R_1 incurs waiting time and while going forward R_0 serves stockpiles on Pad 1 and R_1 serves stockpile on Pad 2, we are interested in the local search neighborhood consisting of all moves that take a single stockpile from M_b^0 and place it in M_f^0 and that take a single stockpile from M_f^1 and place it in M_b^1. We employ a simple descent search, i.e., if such a move improves C_{max}, then then it is accepted, the route of the reclaimer is updated and the search continues. See Algorithm 3 for more details.

Algorithm 3 Restricted ZigZag Routing

Input: The best schedule S obtained after solving an approximation algorithms with Out & Back Routing, where for R_0, we will move a stockpile from the set M_b^0 to the set M_f^0, and for R_1 we will move a stockpile from the set M_f^1 to the set M_b^1.
Output: C_{max}^* and S^*

```
for i ∈ M_b^0 do
    Move the stockpiles a_i^0 from set M_b^0 to M_f^0 at an appropriate place.
    Calculate the value of objective function C^max by serving a_i^0 in the forward move.
    if C_max < C*_max then
        Set C*_max = C_max and S* = S i.e. update the solution value.
        Update the sets M_f^0 and M_b^0.
    end if
    for j ∈ M_f^1 do
        Move the stockpiles a_j^1 from set M_f^1 to M_b^1 at an appropriate place.
        Calculate the value of objective function C^max by serving a_j^1 in the forward move.
        if C_max < C*_max then
            Set C*_max = C_max and S* = S i.e. update the solution value.
            Update the sets M_f^1 and M_f^1.
            break.
        end if
    end for
    if C_max = C*_max then
        break.
    end if
end for
```

4 A COMPUTATIONAL STUDY

To be able to study the performance of the approximation algorithms described above, we implemented a random instance generator requiring the following input parameters: # Stockpiles, % Large Stockpiles, %

Small Stockpiles, Length Range Large Stockpiles, Length Range Small Stockpiles, % Empty Space on Pad, Travel Speed, and Reclaim Speed. For each stockpile on Pad 1, a starting position and a length are generated (uniform randomly) in such a way that no two stockpiles on the pad overlap. The process is repeated for the stockpiles on Pad 2. This is followed by adjustments to the lengths of the stockpiles to obtain the desired % of empty space on the two pads (the adjustments are done in such a way that two pads are of equal lengths).

For our computational experiments the following parameters were used: # Stockpiles = 20, % Large Stockpiles and % Small Stockpiles $\in \{30, 50, 70\}$, Length Range Large Stockpiles = [25,35], Length Range Small Stockpiles = [5,15], % Empty Space on Each Pad $\in \{10, 40\}$, Travel Speed $\in \{2, 8, 20, 100\}$, and Reclaim Speed = 1.

We have generated 10 instances for each combination of parameters. As all algorithms are very efficient on instances of this size, we focus on comparing their performance in terms of quality only. In each of the result tables (Table 1-3), we report the average over the 10 instances of the difference between C^*_{max} and the lower bound provided by the preemptive version of the problem.

Table 1. % Empty Space Pad 1 = 10, % Empty Space Pad 2 = 10

%Large-%Small	Speed	SPLIT	SPLIT$^+$	PARTITION	PARTITION + RZZ	PARTITION + Smart OB
30-70	2	7.255	6.933	6.933	6.342	6.774
	8	5.751	4.315	3.570	2.975	3.411
	20	5.454	3.635	2.467	2.160	2.203
	100	5.292	3.291	0.978	0.926	0.953
50-50	2	6.576	6.576	6.510	6.323	6.407
	8	5.167	3.967	3.566	3.070	3.543
	20	4.872	3.216	2.174	2.022	2.106
	100	4.713	2.846	0.893	0.827	0.893
70-30	2	6.847	6.175	6.048	5.937	5.810
	8	6.074	3.848	3.360	3.034	3.231
	20	5.927	3.497	2.360	2.098	2.161
	100	5.847	3.306	0.928	0.837	0.846

Table 2. % Empty Space Pad 1 = 10, % Empty Space Pad = 40

%Large-%Small	Speed	SPLIT	SPLIT$^+$	PARTITION	PARTITION + RZZ	PARTITION + Smart OB
30-70	2	6.053	5.163	5.163	4.903	4.564
	8	6.139	3.131	2.564	2.542	2.542
	20	5.964	2.710	1.793	1.793	1.793
	100	5.904	2.515	0.803	0.803	0.803
50-50	2	3.472	3.451	3.451	3.451	3.451
	8	3.400	3.098	2.988	2.898	2.898
	20	3.151	2.549	1.748	1.700	1.700
	100	3.114	2.345	0.830	0.826	0.826
70-30	2	4.099	3.419	3.419	3.419	3.419
	8	3.022	3.022	2.669	2.669	2.669
	20	2.751	2.751	2.019	1.850	1.850
	100	2.638	2.577	1.085	0.997	0.997

Table 3. % Empty Space Pad 1 = 40, % Empty Space Pad 2 = 40

%Large-%Small	Speed	SPLIT	SPLIT$^+$	PARTITION	PARTITION + RZZ	PARTITION + Smart OB
30-70	2	6.020	5.280	5.280	5.240	5.240
	8	5.721	3.384	3.237	2.790	2.790
	20	5.734	2.825	2.451	1.948	1.948
	100	5.781	2.698	1.067	0.947	0.947
50-50	2	3.362	3.362	3.362	2.968	2.968
	8	3.321	3.204	2.879	2.639	2.639
	20	3.671	3.208	2.259	2.005	2.005
	100	3.940	3.289	1.030	0.981	0.981
70-30	2	2.429	2.091	2.091	2.091	2.091
	8	2.272	2.272	2.234	2.129	2.129
	20	2.731	2.624	1.802	1.802	1.802
	100	3.210	2.896	1.106	1.022	1.022

The results clearly demonstrate the benefit of evaluating the two possible assignments for the stockpile i that crosses the split point, i.e., **SPLIT**$^+$ performs noticeably better than **SPLIT**$^+$. The results also show the impact of the travel speed of the reclaimers. When the travel speed of the reclaimers gets large (relative to their reclaim speed), **PARTITION**, which is able to exploit an increase in (relative) travel speed, produces schedules that are close to optimal. Finally, we see that careful routing of the reclaimers, i.e., introducing zig-zagging, can

almost always improve upon simple out-and-back routing. Furthermore, introducing zig-zagging appears to be more effective when the amount of reclaiming that needs to be done on both pads is more balanced and when there is less empty space on the pads.

5 FUTURE WORK

In Angelelli et al. (2013), we analyze a number of variants of a basic abstract reclaimer scheduling problem. In this paper, we presented a detailed analysis of one of the more interesting variants, namely one in which two reclaimers need to reclaim a set of stockpiles with given positions on the pads without any precedence requirements between stockpiles. Up to now, we have ignored the dynamic nature of real-life reclaimer scheduling by assuming that all stockpiles are known up front and that all stockpiles fit together on the stock pads. We are currently studying a variant in which the stockpiles do not fit together on the stock pads and a stacking sequence has to be determined as well.

REFERENCES

Angelelli, E., R. Kapoor, and M. Savelsbergh (2013). Complexity results and algorithms for reclaimer scheduling problems. *In preparation*.

Hu, D. and Z. Yao (2012, November). Stacker-reclaimer scheduling in a dry bulk terminal. *Int. J. Comput. Integr. Manuf. 25*(11), 1047–1058.

Kim, K. H. and Y.-M. Park (2004). A crane scheduling method for port container terminals. *European Journal of operational research 156*(3), 752–768.

Kim, K. Y. and K. H. Kim (2003). Heuristic algorithms for routing yard-side equipment for minimizing loading times in container terminals. *Naval Research Logistics (NRL) 50*(5), 498–514.

Lim, A., B. Rodrigues, F. Xiao, and Y. Zhu (2004). Crane scheduling with spatial constraints. *Naval Research Logistics (NRL) 51*(3), 386–406.

Linn, R., J.-y. Liu, Y.-w. Wan, C. Zhang, and K. G. Murty (2003, October). Rubber tired gantry crane deployment for container yard operation. *Comput. Ind. Eng. 45*(3), 429–442.

Liu, J., Y.-w. Wan, and L. Wang (2006). Quay crane scheduling at container terminals to minimize the maximum relative tardiness of vessel departures. *Naval Research Logistics (NRL) 53*(1), 60–74.

Moccia, L., J.-F. Cordeau, M. Gaudioso, and G. Laporte (2006). A branch-and-cut algorithm for the quay crane scheduling problem in a container terminal. *Naval Research Logistics (NRL) 53*(1), 45–59.

Ng, W. and K. Mak (2005a). Yard crane scheduling in port container terminals. *Applied Mathematical Modelling 29*(3), 263 – 276.

Ng, W. C. (2005, July). Crane scheduling in container yards with inter-crane interference. *European Journal of Operational Research 164*(1), 64–78.

Ng, W. C. and K. L. Mak (2005b). An effective heuristic for scheduling a yard crane to handle jobs with different ready times. *Engineering Optimization 37*(8), 867–877.

Ng, W. C. and K. L. Mak (2006). Quay crane scheduling in container terminals. *Engineering Optimization 38*(6), 723–737.

Petering, M. E. (2009). Effect of block width and storage yard layout on marine container terminal performance. *Transportation Research Part E: Logistics and Transportation Review 45*(4), 591–610.

Peterkofsky, R. I. and C. F. Daganzo (1990). A branch and bound solution method for the crane scheduling problem. *Transportation Research Part B: Methodological 24*(3), 159–172.

Young Kim, K. and K. Hwan Kim (1999). A routing algorithm for a single straddle carrier to load export containers onto a containership. *International Journal of Production Economics 59*(1), 425–433.

Zhang, C., Y.-w. Wan, J. Liu, and R. J. Linn (2002). Dynamic crane deployment in container storage yards. *Transportation Research Part B: Methodological 36*(6), 537–555.