Modelling and Optimization of Run-of-Mine Stockpile Recovery

Hirad Assimi
School of Computer Science,
University of Adelaide
Adelaide, South Australia
hirad.assimi@adelaide.edu.au

Ben Koch
EKA Software Solutions
Adelaide, South Australia
ben.koch@ekaplus.com

Chris Garcia
EKA Software Solutions
Adelaide, South Australia
chris.garcia@ekaplus.com

Markus Wagner
School of Computer Science,
University of Adelaide
Adelaide, South Australia
markus.wagner@adelaide.edu.au

Frank Neumann
School of Computer Science,
University of Adelaide
Adelaide, South Australia
frank.neumann@adelaide.edu.au

ABSTRACT

Run-of-Mine stockpiles are essential components in the mining value chain because they can be used as temporary storage to balance inflow and outflow and provide an opportunity for blending material. Stockpile schedulers plan stockpile recovery to balance throughput and material specifications to deliver for the supply chain's next stage. There are technical limits on deliveries where "failure to meet" can lead to significant penalty fees, increased operational costs due to poor operational plans or over-delivery to material specifications. Currently, human experts determine the planning of stockpile recovery in practice. However, this approach is error prone due to the complex distribution of materials within a stockpile and the inability to foresee upcoming deliveries efficiently. In this paper, we model the stockpile recovery problem as a combinatorial optimization problem considering technical restrictions in real-world issues, and we investigate multiple scenarios and experiments. We apply deterministic and randomized greedy algorithms, as well as ant colony optimization algorithms integrated with local search. We compare all algorithms with a rule of thumb heuristic to evaluate our methodology's quality. Our findings show that ant colony optimization outperforms other algorithms, and the variant integrated with swap and insert local search operators finds the best solutions.

CCS CONCEPTS

• **Computing methodologies** → *Randomized search*; *Bio-inspired approaches*;

KEYWORDS

ROM stockpiles, Ant colony optimization, Greedy algorithms, Local search

SAC '21, March 22–26, 2021, Virtual Event, Republic of Korea

ACM ISBN 978-1-4503-8104-8/21/03…$15.00
https://doi.org/10.1145/3412841.3442084

ACM Reference Format:
Hirad Assimi, Ben Koch, Chris Garcia, Markus Wagner, and Frank Neumann. 2021. Modelling and Optimization of Run-of-Mine Stockpile Recovery. In *The 36th ACM/SIGAPP Symposium on Applied Computing (SAC '21), March 22–26, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3412841.3442084

1 INTRODUCTION

Trucks transport mineral materials extracted from mines to Run-of-Mine (ROM) stockpiles, and multiple stockpiles together form a stockyard. To fulfill the next step in the mining value chain upstream, the stockpile scheduler should select a blend of stacked material in ROM stockyard to prepare for delivering with respect to the end-user requirements such as chemical concentration and particle size distribution limits. We refer to this operation as where the scheduler plans how the stockpiles should be reclaimed for deliveries as *stockpile recovery* [7, 9].

Chemical concentrations impose limitations on the target grade quality of the valuable material such as copper and iron in the provided packages to fulfill a request. In addition, undesirable chemical contaminants exist that should be inside of pre-defined ranges for the different deliveries. Consequently, the scheduler typically aims to maintain the target quality grade of deliveries while being as close as possible to the contaminant limits. Another technical requirement is to limit particle size to avoid overwhelming crushing pieces of equipment with excessive material, that is hard to crush.

These technical restrictions are essential in stockyard management. If the scheduler fails to meet these specified requirements in deliveries, significant financial penalty fees are incurred that are proportionate to the degree of violation to be paid to the end-user.

In the stockyard, recovery machinery such as front-end loaders or bucket wheel reclaimers perform reclamation operations. Reclamation operations require that machines move in the stockyard and reclaim cuts from the stockyard. A cut refers to a portion of stacked material in the stockpile where it can vary in size (such as increment of 1000-5000 tonnes of materials) depending on planning resources. These reclaiming operations take time, as machinery moves from one position to another, and thus cost are incurred for the operation.

The task of stockyard management is to schedule the reclaiming operations, i.e. to determine the sequence of reclamation operations for end-user requests. In scheduling, it is essential to maintain

the quality grade of deliveries and to perform stockpile recovery operations considering the operation costs efficiently.

Currently, in practice, human experts determine the reclamation sequence planning often using rules of thumb. For example, they know that high-quality material should be mixed with low-quality material to meet the target quality grade and confine the contaminants as close as possible to the limit range. However, human planning is subject to error due to the complexities in the stockyard and subject to multiple operational restrictions. Moreover, human planning is a limited decision-making procedure where it is hard to foresee upcoming stockpile recovery scheduling requests. As a result, human planning can lead to poor reclaiming sequences where it incurs penalty fees, unexpected losses in practice, and perturbations in stockyard management.

Therefore, for efficient stockyard management, we require an optimal stockyard recovery planning to meet the specifications, technical restrictions, and minimize the operation costs. We also desire to incorporate the capability in efficient planning for multiple deliveries.

1.1 Related Work

Stockpiles are essential components in the supply chain optimization problems in the mining industry, and hence have been studied comprehensively. For example, in the open-pit mine production scheduling, the main aim is to optimize how the material should be extracted from the open-pit mine, while considering mine block models and other restrictions. Stockpiles are used as a component to maximize the net present value and to maintain lower-grade valuable material for future processing [13, 15].

Stockpiles have been also considered as a component in optimization of reclaimer machines job scheduling such as dry bulk handling for coal terminals. However, these studies only focus on the job scheduling of the reclaimer machines, and they consider the stockpiles as a dry bulk component in their model of supply chain optimization [1, 8, 18]. As a drawback, their approaches only consider a weighted average quality for the whole stockpiles, and neglect to investigate the stockpile at a higher resolution considering cuts with different characteristics.

In the context of mining stockpiles, Lu and Myo [11, 12] study stockpile recovery optimization considering cuts in the stockpiles. Their model considers the minimization of the movement by a bucket wheel reclaimer machine. For this purpose, they calculated the movement by Euclidean distance, and they applied mixed integer programming for a small scale of the problem considering two requests and two stockpiles in a single period. However, their approach and model is limited: (1) the calculation of reclaimer machine movement cost in practice is more complicated than a simple Euclidean distance; (2) the stockyard in practice is bigger and the resolution of cuts can determine the size and complexity of the stockpile recovery problem; (3) in planning for deliveries, it is essential to schedule deliveries for a longer period to foresee the future; (4) reclaimer machines also can be other types of machines that can reclaim in different directions; and (5) more technical restrictions exist than a weighted average quality to meet the specified requirements in practice.

We can see that there exists a gap in the literature where it is required to investigate and analyze the stockpile recovery problem and to address real-world features of this problem.

1.2 Our Contribution

In this paper, we consider the stockyard recovery scheduling as a technical challenge in the upstream mining value chain.

We define the problem as a combinatorial optimization problem where the stockyard is a directed graph with precedence constraints connecting the cuts. These precedence constraints impose how the cuts in the stockpiles can be accessed. To address technical restrictions and stockpile management objectives, we introduce a lexicographic objective function for optimization considering the importance of target grade qualities and operation costs where the former has a higher priority.

To simulate the stockyard, we use the information provided by our industrial partner, where they created cuts and stockpiles using real depositing GPS data of trucks in practice. We define multiple scenarios and experiments to represent a stockyard in practice with different complexities of technical restrictions in the problem.

The goal in this study is to model the stockpile recovery problem considering various objective functions and multiple technical restrictions. We desire to develop an approach for efficient scheduling independent of stockyard and reclaimer machine types. As we expand the scenarios in this study, we aim to improve the modelling to be more like a real-world problem.

We intend to find potential solutions where each solution is a reclaiming sequence plan for different scenarios. A good solution should meet the target grade qualities as closely as possible while minimizing the operation cost.

Because we are dealing with a real-world problem, we model the problem and use methodologies so that we can tackle various objective functions, non-linear constraints and further technical restrictions here and in future extensions.

Our search space is constrained by precedence constraints that impose how the cuts can be reclaimed with respect to physical requirements. To deal with these constraints, we develop methodologies that can construct a valid reclaiming sequence step by step. In this manner, we can easily adhere to the precedence constraints, and it can always lead to a valid solution with respect to these constraints.

The algorithms that we consider range from a deterministic and a randomized greedy algorithm (GA) to an ant colony optimization (ACO) algorithm. We also investigate permutation-based local search operators integrated with ACO. All these algorithms construct a solution step by step with a solution construction heuristic appropriate for our objective.

The rest of the paper is organized as follows. In the next section, we define the stockpile recovery combinatorial optimization problem. Next, we introduce the lexicographic objective function and its components. Afterward, we present the scenarios of the problem for investigation. Following, we describe the optimization algorithms and our methodology to deal with the problem. We setup experiments and report on the behavior and quality of obtained solutions and algorithms for different scenarios. We benchmark against a rule of thumb heuristic called the Pilgrim Step Reclaiming

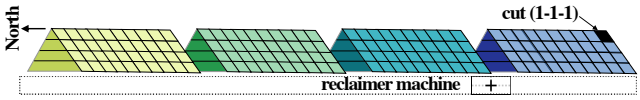

Figure 1: Schematic of the stockyard. Cut (1-1-1) is the entry cut for stockpile recovery where it is the first cut on stockpile 1, top bench and first cut from South-to-North direction.

Heuristic. We observe that the randomization of greedy algorithms can help obtain better solutions in the scenarios, and that ant colony optimization can outperform all algorithms. Finally, we finish with some concluding remarks and some suggestions for future work.

2 PROBLEM STATEMENT

In the following, we define the stockpile recovery optimization problem.

The problem inputs are stockyard structure, reclamation time matrix, information on required deliveries, and available reclaimer machines. We aim to find a sequence of reclamation of cuts in the stockyard to fulfill optimization problem objectives for efficient stockpile management.

The stockyard's structure includes information on multiple stockpiles and information about cuts in each stockpile. We model a stockyard as a directed graph $G = (C, \mathcal{E})$ without cycles, where $C = \{c_1, c_2, \ldots, c_n\}$ is the set of cuts containing n available cuts in the stockyard. Each cut c_j has multiple properties including the percentage of valuable and contaminant elements in the cut denoted by $m_{j,k}$, where $k = 1, 2, \ldots, K$ shows the set of chemical elements. Another property denotes how much material is available in the cut by measuring the tonnage of c_j denoted by Ton_j.

$\mathcal{E}$ is the set of edges connecting cuts and shows the immediate predecessors, where cut i must be reclaimed before j when the reclaimer direction is d and we show a reclamation operation by (i, j, d). Precedence constraints determine the validity of a solution. If it fails for a part of the solution, the solution is impossible to be processed. To perform a reclamation operation, a reclaimer machine can move between two cuts i and j and perform reclamation operation on c_j.

A candidate solution x is a sub-sequence as a permutation of cuts in C, where it represents the order of reclamation to fulfill the optimization problem's objectives. The total time required to perform a reclamation task from c_i to c_j with direction d is given by $T_{i,j,d}$. $\mathcal{T}_d$ shows the reclamation time matrix, which is a full square matrix for all cuts in each direction.

We can identify a cut in the stockyard by its position in the stockyard: each stockpile has benches and a cut is a portion of stacked material in each bench (see Figure 1 for an example).

As mentioned before, it is essential to preserve the quality of deliveries. One objective is the average target quality which aims to preserve the quality of deliveries where the average of chemical contaminants in a delivery should be in a predefined range as follows for a set of chemical elements (K)

$$\hat{m}_{x_k} = \frac{1}{|x|} \sum_{j=1}^{|x|} m_{k,j}$$

Where $\hat{m}_{x_k}$ shows the average of chemical contaminants for cuts in in x with respect to penalty chemical element k.

$$\underline{m_k} \leq \hat{m}_{x_k} \leq \overline{m_k} \quad \forall k \in K$$

Because the chemical properties of cuts have different magnitudes with their corresponding lower and upper bound, we evaluate the degree of violation for target quality using the bracket-operator penalty method $\langle \hat{m}_{x_k} \rangle$ [4], which is calculated as follows:

$$\langle \hat{m}_{x_k} \rangle = \begin{cases} \frac{|\hat{m}_{x_k} - \overline{m_k}|}{|\overline{m_k}|} & \text{if } \hat{m}_{x_k} > \overline{m_k} \\ \frac{|\hat{m}_{x_k} - \underline{m_k}|}{|\underline{m_k}|} & \text{if } \hat{m}_{x_k} < \underline{m_k} \\ 0 & \underline{m_k} \leq \hat{m}_{x_k} \leq \overline{m_k} \end{cases}$$

To calculate the violation for average target quality in x, we use:

$$v_1(x) = \sum_{k=1}^{K} \langle \hat{m}_{x_k} \rangle$$

Another objective in stockyard recovery is the window quality. This objective ensures that when the stockyard scheduler prepares the packages for reclaimed material, the quality of each package should meet another pre-defined window quality. To calculate the violation for deviation of window quality we have:

$$v_2(x) = \sum_{j=4}^{|x|} \left\langle \frac{(m_{j,k} + m_{j-1,k} + m_{j-2,k}}{3}) \right\rangle$$

where j here denotes the position of a cut in a delivery where we desire to look at the window quality after three cuts are already reclaimed for a delivery.

To consider the operation costs, a desirable objective is to have a schedule with ability to reclaim more material in a shorter time which it will result in reducing operation costs. For reclamation operation (i, j, d) we have:

$$u(x) = \sum_{(i,j,d) \in x} \frac{T_{i,j,d}}{Ton_j}$$

The numerator indicates the corresponding element from reclamation time matrix to perform a task and the denominator is the tonnage of cut j available in stockyard information.

2.1 Objective Function

As a stockyard manager, the primary objective is to avoid paying penalty fees with respect to the quality objectives of deliveries. If it is possible to provide a valid solution where all quality objectives are satisfied. The next objective is to reduce the operation costs.[1] To achieve these objectives, we define the objective function for minimization in lexicographic order:

$$f(x) = (v_1(x), v_2(x), u(x))$$

[1] Note that this reflects the preferences of our industry partner. Other companies might aim for different goals.

where order in the objective function matters. To compare tuples of two solutions (x and y):

$$\begin{aligned} & f(x) \leq f(y) \\ \text{iff} \quad & v_1(x) \leq v_1(y) \vee \\ & (v_1(x) = v_1(y) \wedge v_2(x) \leq v_2(y)) \vee \\ & (v_1(x) = v_1(y) \wedge v_2(x) = v_2(y) \wedge u(x) \leq u(y)) \end{aligned}$$

For example, between quality constraints, $v_1(x)$ is more important than $v_2(x)$. The utility function denotes that the manager's objective is reclaiming more material in a shorter time to reduce operating costs.

2.2 Scenarios of the Problem

We define three scenarios for investigation of the optimization problem with different complexities. As we proceed, the problem conditions becomes more similar to the real-world issue in the stockyard management. For each scenario, we consider four experiments for a more detailed investigation. We show an experiment in a scenario by SC X-Y where X and Y refer to the scenario and experiment numbers, respectively.

In the first scenario, we simulate a stockpile recovery problem where we aim to reclaim the whole stockyard. This scenario helps us investigate the quality of algorithms where technical restrictions are loosened, and algorithms can demonstrate their behavior in exploring the stockyard to obtain a low-cost reclaiming sequence. We increase the size of stockyard in experiments, where SC1-1, SC1-2, SC1-3 and SC1-4 denote that the stockyard contains 1, 2, 3, and 4 stockpiles, respectively, and the termination criterion in solution construction is the reclamation of the whole stockyard. In this scenario, we also neglect $v_2(x)$ to make it easier to tackle. For its objective function, we have

$$f_1(x) = (v_1(x), u(x))$$

In the second scenario, we simulate a problem considering technical restrictions in the stockyard. We aim to optimize the reclaiming sequence considering deliveries. We define a sequence for a delivery as x' where $x' \in x$. Similar to the first scenario, we neglect the window quality constraint. We define different experiments, considering the number of deliveries, where SC2-1, SC2-2, SC2-3, SC2-4 show experiments for second scenario where we plan for one, two, three and four deliveries respectively. Moreover, we consider the stockyard as being initially full, and thus include all stockpiles for reclamation. The termination criterion is constructing a solution where it plans for all deliveries in the experiment. We define the objective function as:

$$f_2(x) = \sum_{x' \in x} \big(v_1(x'), u(x')\big)$$

In the third scenario, we make the problem more similar to real-world problem by technical restrictions: all criteria are as same as the second scenario, but we consider the window quality too. Our objective function is

$$f_3(x) = \sum_{x' \in x} \big(v_1(x'), v_2(x'), u(x')\big)$$

The stated problem can be interpreted as an extended version of Travelling Salesperson Problem (TSP) incorporating real-world constraints and objectives. In our problem, cities and distances of TSP are replaced with cuts and the time spent on moving from one cut to another cut and perform a reclamation, respectively. Instead of finding the shortest possible route, we are looking to find a solution to deliver objectives with respect to the end-user requirements. Note that there also exist precedence constraints in accessing the cuts in the graph and some restrictions for preserving target quality.

3 OPTIMIZATION METHODS

In this section, we describe the algorithms applied in this study. We focus on algorithms that construct solutions step by step, considering the precedence constraints. Therefore, the solution is constructed step by step, considering available cuts in the neighborhood to choose as a successor for a solution. We investigate deterministic and randomized greedy algorithms, ant colony optimization, and its variant with local search operators. We also describe a heuristic representing a rule-of-thumb in stockyard management, namely the Pilgrim Step Reclaiming Heuristic (PSRH). We compare the iterative optimization methods with PSRH to evaluate the performance and quality of other algorithms.

3.1 Greedy Algorithm and Randomization

Greedy algorithms are straightforward and fast in constructing a solution for an optimization problem, and they are easy to implement. Algorithm 1 shows the procedure for deterministic greedy algorithm (DGA). DGA starts with an $S = \emptyset$, and at each time step (t), it adds a successor component to x. For the stockpile recovery problem, the entry cut pre-defined by the problem is the primary successor. Next, DGA collects the available cuts (with respect to the precedence constraints) in the initial cut neighborhood as C^t considering the precedence constraints. DGA evaluates the objective function $f(c_j)$ for performing reclamation from the initial cut to the next cut for each cut in $c_j \in C^t$. DGA chooses the successor cut with the highest greediness as

$$c^* = \underset{c_j \in C^t}{\operatorname{argmin}} f(c_j)$$

DGA may choose the best cuts in the stockyard early in the planning. This act of greediness can lead to reclaiming all good material in the stockyard early and getting trapped in local optima to plan deliveries, because the individual deliveries are put together in a strict sequential order. Other studies (e.g. [6]) suggest that controlling the greediness can lead to better solutions.

Randomized Greedy Algorithm (RGA) is a simple randomized version of DGA, which gives it the potential to find better solutions. It has a greedy control parameter $\lambda \geq 0$ where the selection of the successor cut (c^*) occurs according to the probability distribution given by

$$p(c_j|C^t) = \frac{\eta(c_j)^\lambda}{\sum_{c_l \in C^t} \eta(c_l)^\lambda}$$

, where $\eta(c_j)$ denotes the greedy function to make sure that a better candidate has a higher chance to be selected in the neighborhood as follows.

$$\eta(c_j) = \frac{1}{f'(c_j)}$$

Algorithm 1: Deterministic Greedy Algorithm (DGA)

$x := \emptyset$
repeat
  Choose a successor cut $c^* = \text{argmin}_{c_j \in C^t} f(c_j)$
  $x := x \cup c^*$
until *x is complete*
return x

Algorithm 2: Randomized Greedy Algorithm (RGA)

$x := \emptyset$
repeat
  Choose a successor cut (c^*) according to probability
  $p(c_j|C^t) = \dfrac{\eta(c_j)^\lambda}{\sum_{c_l \in C^t} \eta(c_l)^\lambda}$
  $x := x \cup c^*$
until *x is complete*
return x

where $f'(c_j)$ is a mapped real number of $f(c_j)$ required for selection procedure among cuts in the neighborhood. We classify all cuts in the neighborhood in three sets ($\mathcal{S}$). Set 1 contains valid solutions, set 2 include solutions with $v_1(c_j) = 0 \wedge v_2(c_j) \neq 0$ and other solutions are put in set 3. For each set separately, we normalize the objective functions component wise to $(0, 1)$ by a linear mapping and we denote it by $\mathcal{N}(c_j)$:

$$\mathcal{N}(c_j) = \mathcal{N}\left(u(c_j)\right) + \mathcal{N}\left(v_2(c_j)\right) + \mathcal{N}\left(v_1(c_j)\right)$$

to calculate $f'(c_j)$, we use:

$$f'(c_j) = \begin{cases} \mathcal{N}(c_j) + 1 & \text{if } c_j \in \mathcal{S}_1 \\ \mathcal{N}(c_j) + 1 + 10 & \text{if } c_j \in \mathcal{S}_2 \\ \mathcal{N}(c_j) + 1 + 10 + 100 & \text{Otherwise.} \end{cases}$$

Note that the addition of 1 ensures that $p(c_j|C^t) > 0$ and that the addends of 10 and 100 helps to partially preserve the order of objective function. Our probabilistic selection represents a roulette wheel selection where the chance of selecting a candidate in C^t is proportionate to its fitness. λ determines how greedy RGA acts for the selection procedure: as $\lambda \to \infty$, RGA approaches the behavior of DGA. Algorithm 2 shows the procedure for RGA.

3.2 Ant Colony optimization (ACO)

Ant Colony Optimization (ACO) is swarm-intelligence approach that uses artificial ants [5], which has been inspired by the foraging behavior of ants in nature. Some ants have been observed to perform random walk to find a source of food. On their return route to the colony, they deposit pheromone and other ants can sense it and identify a good route instead of a random walk. More ants follow a route, depositing more pheromone, and thus reinforce the route. In addition, pheromone evaporates gradually to reduce the attraction capability of untraveled edges. ACO uses artificial ants to simulate this foraging and it has been successfully applied to combinatorial optimization and open-pit mining scheduling problem [2, 14, 16]. Importantly for us here, ACO can be viewed as an iterative and adaptive RGA with more control parameters.

Algorithm 3: Ant Colony optimization (ACO)

initialize τ ▷ pheromone values initialization
generate ants for initial colony $\pi_i \subset \Pi$
repeat
  for *each ant* π_i **do**
    repeat
      construct a solution x step by step, probabilistically
    until *solution is complete*
  for *each ant* $\in \Pi^*$ **do**
    perform local search ▷ optional
  Update best found solution x^*
  Update τ
until *ACO termination criterion met*
return x^*

Algorithm 3 shows the procedure of ACO. First, ACO initializes the pheromone matrix for the initial generation and generates a colony contains artificial ants. Each ant performs a random walk to construct a solution step by step. The probability distribution that each ant uses to choose a successor (i.e. the next cut) while constructing a solution is

$$p(c_j|C^t) = \frac{[\tau_{i,j,d}]^\alpha [\eta(c_j)]^\beta}{\sum_{c_l \in C^t} [\tau_{i,l,d}]^\alpha [\eta(c_l)]^\beta}$$

where parameters α and β are control parameters in selection to balance using pheromone or heuristic information. After all ants finish their independent random walk, it is optional for ACO to employ a local search on the found solutions. At the end of the generation, ACO updates the pheromone values and its best found solution. This procedure repeats until ACO's termination criterion is met.

For this study, we use a variant of ACO, namely Max-Min Ant System (MMAS) [17]. In MMAS, only the best ant at each generation deposits pheromone to reinforce its solution. The pheromone range is limited to avoid becoming very big or very small. This feature can prevent getting trapped in local optimal.

To initialize the pheromone matrix (τ), we consider equal amount of pheromone on all edges in $\mathcal{E}$: $\tau_{i,j,d} = 1/2$. We restrict each $\tau_{i,j,d}$ in the interval $\left[\frac{1}{|C|}, 1 - \frac{1}{|C|}\right]$ [14]. After all ants construct their solution, the best ant at the iteration (with the best obtained solution at iteration x') deposits pheromone on the its solution edges as follows,

$$\tau'_{i,j,d} = \begin{cases} \min\{(1-\rho).\tau_{i,j,d} + \rho, 1 - \frac{1}{|C|}\} & \text{if } (i,j,d) \in x' \\ \max\{(1-\rho).\tau_{i,j,d}, \frac{1}{|C|}\} & \text{otherwise.} \end{cases}$$

where $0 < \rho < 1$ denotes the evaporation factor.

3.3 ACO with Local Search

Local search can be complimentary to ACO, because ACO explores the search space coarsely. However local search can help to explore in the neighborhood of a constructed solution more finely. Therefore, after ants construct a solution, their obtained solution can be

a good starting point for local search. We refer interested readers to [10] for more information.

In this paper, we employ three well-known operators for permutation search problems namely *swap*, *insert* and *inverse* operators. Considering one solution x, these operators get two components in x and perform a local search on the components' position. The swap operator exchanges two components of the solution; the insert operator shifts the second component ahead of the first component. The inverse operator arranges all components between and including the two components in the opposite order.

We investigate the iterative local search in the neighborhood of all components. We call variants of MMAS integrated with local search operators as MMAS-swp, MMAS-ins, MMAS-inv for swap, insert and inverse operators, respectively.

3.4 Pilgrim Step Reclaiming Heuristic (PSRH)

To evaluate the above mentioned algorithms in stockpile recovery, we employ a heuristic representing a rule of thumb that is used for manual planning of the reclaiming sequence in practice. Pilgrim Step Reclaiming Heuristic (PSRH) reclaims one stockpile completely from one end to another. First, it reclaims the top bench partially, then the lower bench for the reclaimed cut and reclamation repeats till one end is completely reclaimed. Next, the reclaimer machine proceeds and reclaims the rest of the stockpile alike to previous steps from the top bench to the bottom. For example, the following sequence follows PSRH for the first stockpile:

$$\{(1\text{-}1\text{-}1), (1\text{-}2\text{-}1), (1\text{-}3\text{-}1), (1\text{-}4\text{-}1), (1\text{-}1\text{-}2), \ldots, (1\text{-}4\text{-}2), \ldots, (1\text{-}4\text{-}10)\}$$

PSRH reclaims one stockpile after another, and this unique movement avoids moving the reclaimer machine from one stockpile to another before the complete reclamation of a stockpile. Due to this behavior, the time required to perform tasks is reasonably small and is an efficient heuristic considering the reclamation time and cost. However, the PSRH way of reclaiming can lead to solutions where the constraint violations $v_1(x)$ and $v_2(x)$ are non-zero.

We use PSRH as a reference in practice where we can compare and evaluate the quality of obtained reclaiming plans created by our algorithms.

4 EXPERIMENTAL SETUP

In this section, we describe our experimental setup and the assumptions. We consider a real-world stockyard model provided by our industrial partner. This stockpile has been created using the GPS information of trucks while they stacked the stockpile with real material. The stockyard includes four stockpiles where each has four benches and ten cuts forming a stockyard with 160 cuts.

We identify a cut by ($id_{\text{stockpile}}$-id_{bench}-id_{cut}) where, cut (1-1-1) shows the first cut in first bench of first stockpile and denotes the entry point (see again Figure 1). There is one machine that can reclaim cuts in one direction, which is South to North. This means that a machine can reclaim only in one direction, but it can move back and forth to position itself for the next reclamation.

The machine can only process one reclamation task at a time, and the task should be completed before performing the next reclaiming task. For scenarios 2 and 3, we assume that the required tonnage for each delivery is 100,000 tonnes.

The machine starts from the entry cut for reclamation and moves to the next available cut with respect to precedence constraints. A candidate solution sub-sequence is a permutation of cuts in the stockyard where it shows the order of reclamation to satisfy constraints and deliver the scenario objectives efficiently. A straightforward example of reclamation could be starting from the entry point and reclaim all the cuts in the first bench of the first stockpile. For this reclamation order, the solution is as follows,

$$\{(1\text{-}1\text{-}1), (1\text{-}1\text{-}2), (1\text{-}1\text{-}3), (1\text{-}1\text{-}4), \ldots, (1\text{-}1\text{-}10)\}$$

DGA is deterministic, and there is no parameter to configure. However, RGA has the parameter λ to control the amount of greediness of the algorithm. We set $\lambda = \{1, 3, 5, 7, 10, 15, 20\}$ and we name the RGA variants as RGA-λ. For MMAS, we set $\alpha = 1$, $\beta = 2$, $\rho = 0.5$ and Π^* only contains best ant in the iteration. We also consider ten ants, and the termination criterion for ACO is when 1000 generations elapsed. With respect to the scenario, a solution construction continues until the solution is complete for all algorithms.

For the randomized algorithms, we run each for 50 times to evaluate them fairly. We also carry out statistical comparisons for randomized algorithms by the Kruskal-Wallis test with a 95% confidence interval integrated with the posteriori Bonferroni test for pair-wise comparisons [3]. We rank the obtained solutions by the objective function's lexicographic order to perform the statistical test, and we use their ranks for the statistical test.

For a closer look, we report the median, best and worst solutions obtained by the algorithms in corresponding tables. Note that for PSRH and DGA as they are deterministic algorithms, we report the same value. We also evaluate algorithms by success rate. This measure is the percentage of success for algorithms in obtaining valid solutions. While for deterministic algorithms it is 0 or 1, we calculate the fraction out of 50 runs for randomized algorithms.

5 RESULTS AND DISCUSSION

Figure 2 provides a summary of the results by showing the significance for all scenarios and experiments. Among other, we observe that RGA and MMAS behave differently, while within each group the results are not always different.

5.1 Scenario 1

Here, we assume that the problem aim is reclaiming the whole stockyard. Table 1 lists the obtained objectives for Scenarios 1 and 2. Note that we only report the best RGA variant for each experiment for brevity.

For all experiments, we observe that all algorithms can obtain a success rate of 100%. Therefore, for the evaluation, we only look at the utility cost. We can also see that DGA outperforms PSRH, meaning that acting deterministic greedy is better than using PSRH for this scenario. We observe the same pattern for RGA, but with there is a statistical difference for various experiments. We can see for all experiments that RGA-15 and RGA-20 are not significantly different.

RGA-1, RGA-3, RGA-5 can find solutions with large utility, as the worst obtained utility among these variants are 25.5281, 25.1717, 24.8425, respectively. We see that as λ increases, the median value of utility decreases, where it leads to better solutions. We see for

Table 1: Fitness values obtained for the optimized solutions in Scenarios 1 and 2

Instance		PSRH	DGA	best RGA	MMAS	MMAS-swp	MMAS-ins	MMAS-inv
SC1-1	Median	(0.0, 24.6291)	(0.0, 23.685)	(0.0, 24.0001)	(0.0, 23.1543)	**(0.0, 23.0988)**	(0.0, 23.1042)	**(0.0, 23.0988)**
	Best	(0.0, 24.6291)	(0.0, 23.685)	(0.0, 23.3787)	(0.0, 23.1502)	**(0.0, 23.0472)**	(0.0, 23.0504)	**(0.0, 23.0472)**
	Worst	(0.0, 24.6291)	(0.0, 23.685)	(0.0, 24.8425)	(0.0, 23.1661)	**(0.0, 23.1021)**	(0.0, 23.1535)	(0.0, 23.1229)
	Success rate	1	1	1	1	1	1	1
SC1-2	Median	(0.0, 49.6518)	(0.0, 48.1309)	(0.0, 48.1309)	(0.0, 46.6986)	(0.0, 46.4339)	(0.0, 46.6783)	**(0.0, 46.4335)**
	Best	(0.0, 49.6518)	(0.0, 48.1309)	(0.0, 47.7301)	(0.0, 46.598)	(0.0, 46.4005)	(0.0, 46.5274)	**(0.0, 46.3633)**
	Worst	(0.0, 49.6518)	(0.0, 48.1309)	(0.0, 48.3392)	(0.0, 46.8059)	(0.0, 46.4774)	(0.0, 46.7635)	**(0.0, 46.4756)**
	Success rate	1	1	1	1	1	1	1
SC1-3	Median	(0.0, 74.6085)	(0.0, 72.1901)	(0.0, 72.4698)	(0.0, 70.0757)	**(0.0, 69.7819)**	(0.0, 70.0914)	(0.0, 69.7826)
	Best	(0.0, 74.6085)	(0.0, 72.1901)	(0.0, 71.8263)	(0.0, 69.8758)	**(0.0, 69.501)**	(0.0, 69.921)	(0.0, 69.54)
	Worst	(0.0, 74.6085)	(0.0, 72.1901)	(0.0, 76.5797)	(0.0, 70.289)	**(0.0, 69.9158)**	(0.0, 70.2281)	(0.0, 69.9478)
	Success rate	1	1	1	1	1	1	1
SC1-4	Median	(0.0, 99.5444)	(0.0, 96.2366)	(0.0, 96.6943)	(0.0, 93.7738)	**(0.0, 93.1774)**	(0.0, 93.8173)	(0.0, 93.2369)
	Best	(0.0, 99.5444)	(0.0, 96.2366)	(0.0, 95.8881)	(0.0, 93.4327)	(0.0, 92.9711)	(0.0, 93.4237)	**(0.0, 92.9587)**
	Worst	(0.0, 99.5444)	(0.0, 96.2366)	(0.0, 106.4864)	(0.0, 94.3409)	(0.0, 93.497)	(0.0, 94.1342)	**(0.0, 93.4529)**
	Success rate	1	1	1	1	1	1	1
SC2-1	Median	(0.0, 15.9225)	(0.0, 17.4546)	(0.0, 19.4966)	**(0.0, 15.2188)**	(0.0, 17.2239)	(0.0, 15.2251)	(0.0, 17.1897)
	Best	(0.0, 15.9225)	(0.0, 17.4546)	(0.0, 15.4309)	(0.0, 15.1747)	(0.0, 15.1392)	**(0.0, 15.0877)**	**(0.0, 15.0877)**
	Worst	(0.0, 15.9225)	(0.0, 17.4546)	(0.0, 29.3699)	**(0.0, 15.2474)**	(0.0, 19.6106)	(0.0, 15.2504)	(0.0, 19.1453)
	Success rate	1	1	1	1	1	1	1
SC2-2	Median	(0.0, 34.7849)	(0.0, 41.1948)	(0.0, 37.5494)	(0.0, 31.8562)	(0.0, 34.427)	**(0.0, 31.8151)**	(0.0, 34.4242)
	Best	(0.0, 34.7849)	(0.0, 41.1948)	(0.0, 31.2005)	(0.0, 31.218)	(0.0, 33.5467)	**(0.0, 31.0527)**	(0.0, 33.4021)
	Worst	(0.0, 34.7849)	(0.0, 41.1948)	(0.0, 48.4085)	(0.0, 36.0702)	(0.0, 35.6118)	(0.0, 35.8257)	**(0.0, 35.3716)**
	Success rate	1	1	1	1	1	1	1
SC2-3	Median	(0.0304, 49.6518)	(0.1094, 65.5448)	(0.0743, 61.5351)	(0.0, 51.3597)	(0.0, 51.2627)	(0.0, 51.3262)	**(0.0, 51.1625)**
	Best	(0.0304, 49.6518)	(0.1094, 65.5448)	(0.0, 49.9985)	(0.0, 48.9855)	(0.0, 48.906)	(0.0, 49.4875)	**(0.0, 48.3031)**
	Worst	(0.0304, 49.6518)	(0.1094, 65.5448)	(0.1953, 64.7137)	(0.0, 52.7509)	**(0.0, 52.2798)**	(0.0, 52.6403)	(0.0, 52.403)
	Success rate	0	0	0.36	1	1	1	1
SC2-4	Median	(0.1297, 65.5358)	(0.2492, 85.2887)	(0.135, 93.8736)	(0.0, 67.107)	**(0.0, 66.7176)**	(0.0, 67.1275)	(0.0, 67.0235)
	Best	(0.1297, 65.5358)	(0.2492, 85.2887)	(0.0, 72.2375)	(0.0, 65.3268)	**(0.0, 63.1248)**	(0.0, 64.164)	(0.0, 64.8839)
	Worst	(0.1297, 65.5358)	(0.2492, 85.2887)	(0.276, 93.0093)	(0.0, 68.4933)	(0.0, 68.0874)	**(0.0, 68.0108)**	(0.0, 68.0841)
	Success rate	0	0	0.1	1	1	1	1

experiments 2-4, RGA-20 can obtain the best solution with the utility of 47.7301, 71.8263, and 95.8881, respectively. However, for experiment 1, which is the most straightforward instance in our study, RGA-5 obtains the best solution with a utility of 23.3787, but with a higher median than the RGA variant where $\lambda \geq 7$.

We also observe that the best solution obtained by RGA is the same as DGA in experiment 1. However, as the stockyard's size becomes larger, adding randomness results in RGA outperforming DGA. To determine the best RGA variant of experiments 2-4, we report RGA-20, and for experiment 1. and we report RGA-5 for experiment 1.

We see that all MMAS variants with or without local search outperform PSRH, DGA and RGA variants. For experiment 1, the local search variants outperform those without. Moreover, there is no significant difference among the local search variants. However, for experiments 2-4, we see MMAS-swp and MMAS-ins are significantly different and better than other variants. Nonetheless, there exists no significant difference between MMAS and MMAS-inv.

5.2 Scenario 2

In this scenario, we aim to plan the reclamation to fulfill multiple deliveries. We observe that PRSH and DGA can obtain zero violation solutions for the first two experiments. However, for the last two experiments, their obtained solutions violate the constraints. Moreover, DGA obtains worse solutions than PSRH. For this scenario, we can see that acting greedy by DGA leads to weaker solutions, unlike scenario 1 when the problem is more similar to real-world conditions.

For all experiments, among RGA variants, we see no significant difference where $3 \leq \lambda \leq 10$. Moreover, in experiment 1, we observe that the RGA-1 success rate is 94%; however, other RGA variants are 100% successful. For experiment 1, we report RGA-10, where it can obtain the best solution among RGA variants. For experiment 2, we see that RGA-1, RGA-3, RGA-7, RGA-15, and RGA-20 success rates are at least 90%. Among RGA-10, RGA-15, and RGA-20 where all are non-significant different from each other, we report RGA-10 because its success rate is 100% and it can obtain reasonably good solutions. Experiments 3 and 4 are more challenging instances because we can see that as the number of deliveries increases, the resources in the stockyard are close to being exhausted. It makes it

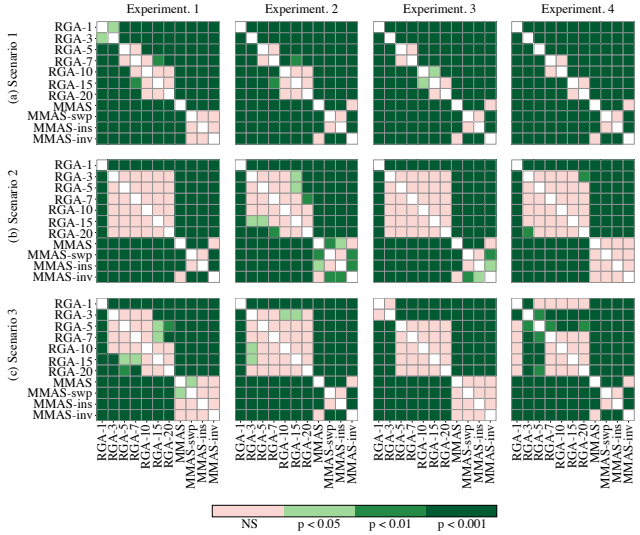

Figure 2: Significance plot of statistical test for randomized algorithms. p denotes the p-value and NS refers to no significant difference.

harder for the algorithm to provide non-zero violations for all the deliveries. For experiment 3, we observe that among RGA variants, as λ increases, the success rate decreases from 80% to 8%. We report RGA-10 as the best RGA variant for experiment 3. For experiment 4, the success rate for RGA variants decreases from 34% to 0%, where RGA-15 success rate is zero. We report RGA-5 as the best RGA variant where it can find the best solution (0.0, 72.2375).

Similar to experiments 2-4 in scenario 1, we see no significant difference between MMAS and MMAS-inv. We also observe that MMAS with or without local search outperforms PSRH, DGA and RGA variants. The success rate for all MMAS variants is 100%, where it enables us to only look at their utility function to compare them more easily. We can see that in experiments 1 and 2, MMAS-ins outperforms MMAS-swp. However, in experiments 3 and 4, with more complicated instances, MMAS-swp is the best among the two. However, in experiment 4, as the most challenging experiment in this scenario, all MMAS variants are not significantly different.

5.3 Scenario 3

This scenario is the most challenging one and the most similar to the real-world application which considers window quality. Table 2 lists the results. In all experiments, we observe that PSRH and DGA obtain valid solutions, where DGA can obtain the best solutions. Among RGA variants for experiments 1-3, we see that variants with $3 \leq \lambda \leq 7$ are non-significant different from each other, and the same behavior applies to the variants with $10 \leq \lambda \leq 20$.

RGA-3 and RGA-5 obtain solutions with a higher partial success rate. Similar to other scenarios, MMAS and its variants outperform other algorithms, and for experiments 1-3, all show a 100% success rate. However, for experiment 4, we see an occasional success for only the MMAS variants. This could be because of the way the experiments were defined: almost all resources are exhausted, and it is hard to find a solution without violating the quality objectives.

For experiment 1, RGA-10, RGA-15 and RGA-20 obtain their best solutions with zero violation with utility of 17.0583, 17.1489 and 16.714 with algorithm success rates of 68%, 44% and 42% respectively. Similarly, For experiment 2, RGA-5, RGA-7, RGA-20 obtain their best solutions as zero-violation with utility of 34.3946, 34.6121 and 34.332 with their success rates as 42%, 20% and 2% respectively. It shows that for these complex scenarios in both experiments, some RGA variants are able to find better solutions than MMAS but with a lower success rate. It shows that there exists room for improvement of MMAS variants with a proper tuning of MMAS and improve MMAS ability to explore the search space better. For experiment 1-3, RGA-20 is the best RGA variant.

Experiment 4 of this scenario is the most difficult instance for our study. We can see that all RGA variants are unsuccessful. However, all MMAS variants can obtain a valid solution but note that the success rate of MMAS is low. It demonstrates that MMAS and its variants can find a good solution. However, the algorithm's quality for robustness in finding valid solutions in multiple runs could be improved. For experiment 4, we RGA-3 is the best RGA variant. It can obtain a solution where $v_1(x) = 0$ as (0.0, 0.3736, 84.114) and it is the best solution among all RGA variants.

6 CONCLUSIONS

In this paper, we have modeled the stockpile recovery problem as a combinatorial optimization problem with a lexicographic objective function considering technical restrictions in practice. We have used optimization methods to construct a reclaiming sequence plan step by step to meet the precedence constraints in solving the problem. For this purpose, we have explored the use of deterministic and randomized greedy algorithms. We have also employed MMAS as a variant of the ACO algorithm, and we considered three local search operators for it, namely swap, insert, and inversion.

In our experiments, which have covered a variety of real-world aspects, we have observed that adding randomness to the deterministic greedy algorithm has helped it find better solutions when the scenario is more similar to the real world problem. We have also seen that MMAS – especially with a local search – can outperform other algorithms with a high success rate.

In this initial study, we have used a number of default parameters for the algorithms. For future studies, it could be interesting (1) to investigate automated tuning of the algorithms, (2) to study other ACO variants, and (3) to develop local search operators that are specific to this reclamation problem.

ACKNOWLEDGMENTS

This research has been supported by the SA Government through the PRIF RCP Mining Consortium.

REFERENCES

[1] Enrico Angelelli, Thomas Kalinowski, Reena Kapoor, and Martin WP Savelsbergh. 2016. A reclaimer scheduling problem arising in coal stockyard management. *Journal of Scheduling* 19, 5 (2016), 563–582.
[2] Jonatas B.C. Chagas and Markus Wagner. 2020. Ants can orienteer a thief in their robbery. *Operations Research Letters* 48, 6 (2020), 708 – 714.
[3] Gregory W Corder and Dale I Foreman. 2014. *Nonparametric statistics: A step-by-step approach.* John Wiley & Sons.
[4] Kalyanmoy Deb. 2001. *Multi-objective optimization using evolutionary algorithms.* Vol. 16. John Wiley & Sons.

Table 2: Objective functions obtained for the optimized solutions in Scenario 3

Instance		PSRH	DGA	best RGA	MMAS	MMAS-swp	MMAS-ins	MMAS-inv
SC2-1	Median	(0.0, 0.0801, 15.9225)	(0.0, 0.0033, 31.9449)	(0.0, 0.0, 26.1112)	**(0.0, 0.0, 21.6272)**	(0.0, 0.0, 22.5023)	(0.0, 0.0, 22.3799)	(0.0, 0.0, 22.4573)
	Best	(0.0, 0.0801, 15.9225)	(0.0, 0.0033, 31.9449)	**(0.0, 0.0, 16.714)**	(0.0, 0.0, 17.5283)	(0.0, 0.0, 20.5082)	(0.0, 0.0, 20.7674)	(0.0, 0.0, 19.6313)
	Worst	(0.0, 0.0801, 15.9225)	(0.0, 0.0033, 31.9449)	(0.0, 0.0, 27.6398)	**(0.0, 0.0, 22.8708)**	(0.0, 0.0, 23.553)	(0.0, 0.0, 23.0829)	(0.0, 0.0, 23.393)
	Success rate	0	0	0.42	1	1	1	1
SC2-2	Median	(0.0, 0.2286, 34.7849)	(0.0, 0.0668, 54.3735)	(0.0, 0.0, 43.803)	(0.0, 0.0, 38.6463)	**(0.0, 0.0, 37.7133)**	(0.0, 0.0, 38.5856)	(0.0, 0.0, 37.966)
	Best	(0.0, 0.2286, 34.7849)	(0.0, 0.0668, 54.3735)	**(0.0, 0.0, 34.332)**	(0.0, 0.0, 35.2312)	(0.0, 0.0, 36.1584)	(0.0, 0.0, 36.916)	(0.0, 0.0, 36.012)
	Worst	(0.0, 0.2286, 34.7849)	(0.0, 0.0668, 54.3735)	(0.062, 0.0, 47.5242)	(0.0, 0.0, 41.0108)	(0.0, 0.0, 40.1809)	(0.0, 0.0, 39.9312)	**(0.0, 0.0, 39.6312)**
	Success rate	0	0	0.02	1	1	1	1
SC2-3	Median	(0.0304, 0.4242, 49.6518)	(0.0, 0.1015, 70.2569)	(0.0496, 0.0489, 63.8654)	(0.0, 0.0, 59.6043)	**(0.0, 0.0, 55.9089)**	(0.0, 0.0, 58.6546)	(0.0, 0.0, 56.0387)
	Best	(0.0304, 0.4242, 49.6518)	(0.0, 0.1015, 70.2569)	(0.0, 0.0489, 57.0279)	(0.0, 0.0, 56.3588)	**(0.0, 0.0, 53.7049)**	(0.0, 0.0, 56.568)	(0.0, 0.0, 54.2217)
	Worst	(0.0304, 0.4242, 49.6518)	(0.0, 0.1015, 70.2569)	(0.1295, 0.0489, 79.1781)	(0.0, 0.0, 62.2242)	**(0.0, 0.0, 57.2936)**	(0.0, 0.0, 59.7979)	(0.0, 0.0, 57.5471)
	Success rate	0	0	0	1	1	1	1
SC2-4	Median	(0.1297, 2.0262, 65.5358)	(0.0627, 0.7477, 84.094)	(0.126, 0.3736, 82.627)	(0.0, 0.0064, 81.1742)	(0.0, 0.0, 80.9867)	(0.0, 0.0281, 79.3744)	**(0.0, 0.0, 76.3152)**
	Best	(0.1297, 2.0262, 65.5358)	(0.0627, 0.7477, 84.094)	(0.0, 0.3736, 84.114)	(0.0, 0.0064, 88.1848)	(0.0, 0.0, 81.6204)	(0.0, 0.0281, 82.8373)	**(0.0, 0.0, 72.6274)**
	Worst	(0.1297, 2.0262, 65.5358)	(0.0627, 0.7477, 84.094)	(0.2251, 0.3736, 97.6799)	(0.0, 0.0064, 80.9308)	**(0.0, 0.0, 77.0153)**	(0.0, 0.0281, 82.1064)	(0.0, 0.0, 83.4832)
	Success rate	0	0	0	0	0.02	0	0.1

[5] Marco Dorigo and Thomas Stützle. 2004. *Ant colony optimization*. Cambridge, MA: MIT Press.

[6] Wanru Gao, Tobias Friedrich, Frank Neumann, and Christian Hercher. 2018. Randomized greedy algorithms for covering problems. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 309–315.

[7] K. Jupp, T. J. Howard, and J. E. Everett. 2013. Role of pre-crusher stockpiling for grade control in iron ore mining. *Applied Earth Science* 122, 4 (2013), 242–255.

[8] Thomas Kalinowski, Reena Kapoor, and Martin WP Savelsbergh. 2017. Scheduling reclaimers serving a stock pad at a coal terminal. *Journal of Scheduling* 20, 1 (2017), 85–101.

[9] Siyi Li, Marco de Werk, Louis St-Pierre, and Mustafa Kumral. 2019. Dimensioning a stockpile operation using principal component analysis. *International Journal of Minerals, Metallurgy and Materials* 26, 12 (2019), 1485–1494.

[10] Helena R Lourenço, Olivier C Martin, and Thomas Stützle. 2003. Iterated local search. In *Handbook of metaheuristics*. Springer, 320–353.

[11] Tien-Fu Lu and Maung Thi Rein Myo. 2010. Optimization of reclaiming voxels for quality grade target with reclaimer minimum movement. In *2010 11th International Conference on Control Automation Robotics & Vision*. IEEE, 341–345.

[12] Tien-Fu Lu and Maung Thi Rein Myo. 2011. Optimal stockpile voxel identification based on reclaimer minimum movement for target grade. *International Journal of Mineral Processing* 98, 1-2 (2011), 74–81.

[13] Eduardo Moreno, Mojtaba Rezakhah, Alexandra Newman, and Felipe Ferreira. 2017. Linear models for stockpiling in open-pit mine production scheduling problems. *European Journal of Operational Research* 260, 1 (2017), 212–221.

[14] Frank Neumann, Dirk Sudholt, and Carsten Witt. 2009. Analysis of different MMAS ACO algorithms on unimodal functions and plateaus. *Swarm Intelligence* 3, 1 (2009), 35–68.

[15] Mojtaba Rezakhah, Eduardo Moreno, and Alexandra Newman. 2020. Practical performance of an open pit mine scheduling model considering blending and stockpiling. *Computers & Operations Research* 115 (2020), 104638.

[16] Masoud Soleymani Shishvan and Javad Sattarvand. 2015. Long term production planning of open pit mines by ant colony optimization. *European Journal of Operational Research* 240, 3 (2015), 825–836.

[17] Thomas Stützle and Holger H. Hoos. 2000. MAX–MIN Ant System. *Future Generation Computer Systems* 16, 8 (2000), 889 – 914.

[18] Ozgur Unsal and Ceyda Oguz. 2019. An exact algorithm for integrated planning of operations in dry bulk terminals. *Transportation Research Part E: Logistics and Transportation Review* 126 (2019), 103–121.