Les Cahiers du GERAD ISSN: 0711–2440

Coordination of electrical drilling machines in open-pit mines: A constraint programming approach

M. Maftah, M. Gamache, B. Agard, M. Gendreau

G–2024–62

September 2024

La publication de ces rapports de recherche est rendue possible grâce au soutien de HEC Montréal, Polytechnique Montréal, Université McGill, Université du Québec à Montréal, ainsi que du Fonds de recherche du Québec – Nature et technologies.

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2024
– Bibliothèque et Archives Canada, 2024

The publication of these research reports is made possible thanks to the support of HEC Montréal, Polytechnique Montréal, McGill University, Université du Québec à Montréal, as well as the Fonds de recherche du Québec – Nature et technologies.

Legal deposit – Bibliothèque et Archives nationales du Québec, 2024
– Library and Archives Canada, 2024

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

Coordination of electrical drilling machines in open-pit mines: A constraint programming approach

Mohamed Maftah [a, b]

Michel Gamache [a, b]

Bruno Agard [a]

Michel Gendreau [a]

[a] *Department of Mathematics and Industrial Engineering, Polytechnique Montréal, Montréal (Qc), Canada, H3T 1J4*

[b] *GERAD, Montréal (Qc), Canada, H3T 1J4*

mohamed.maftah@polymtl.ca
michel.gamache@polymtl.ca
bruno.agard@polymtl.ca
michel.gendreau@polymtl.ca

September 2024
Les Cahiers du GERAD
G–2024–62

Abstract : This article addresses the efficient coordination of multiple electrical drill rigs in open-pit mines, considering unique constraints such as continuous power supply, dynamic obstacle creation, and safe distancing. We aim to maximize drilling within a given time horizon. We provide a comprehensive problem description and propose a constraint programming model complemented by a heuristic algorithm. Experimental results demonstrate the model's effectiveness in scheduling up to 300 tasks with 3 machines over 24 hours, achieving near-optimal solutions within 2 minutes. For instances with fewer than 250 tasks, the model consistently reaches optimal solutions, enabling real-time decision-making in dynamic mining environments.

Keywords: Open-pit mining, electrical drill rigs, scheduling, constraint programming, collision avoidance, real-time decision-making

Résumé : Cet article aborde la coordination efficace de multiples foreuses électriques dans les mines à ciel ouvert, en tenant compte de contraintes uniques telles que l'alimentation électrique continue, la création d'obstacles dynamiques et la distanciation sécuritaire. Notre objectif est de maximiser le forage dans un horizon temporel donné. Nous fournissons une description complète du problème et proposons un modèle de programmation par contraintes complété par un algorithme heuristique. Les résultats expérimentaux démontrent l'efficacité du modèle pour planifier jusqu'à 300 tâches avec 3 machines sur 24 heures, atteignant des solutions quasi-optimales en moins de 2 minutes. Pour les instances de moins de 250 tâches, le modèle atteint systématiquement des solutions optimales, permettant une prise de décision en temps réel dans des environnements miniers dynamiques.

Mots clés : Mines à ciel ouvert, foreuses électriques, ordonnancement, programmation par contraintes, évitement de collisions, prise de décision en temps réel

Acknowledgements: This work was supported by the NSERC Cooperative Research Development program under Grant RDCPJ 531815018.

1 Introduction

Open-pit mining operations have undergone significant evolution in recent decades, driven by technological advancements and the increasing demand for efficient resource extraction (Barnewold & Lottermoser, 2020). At the core of these operations lies a complex interplay of activities: drilling, blasting, loading, and hauling (Mwangi Akisa, Huang, Jianhua, Kasomo, & Matidza, 2021). While each of these processes has been subject to extensive research and optimization efforts (Osanloo, Gholamnejad, & Karimi, 2008; Fathollahzadeh, Asad, Mardaneh, & Cigla, 2021; Alarie & Gamache, 2002; Moradi Afrapoli & Askari-Nasab, 2019), the drilling phase, in particular, presents unique challenges and opportunities for improvement that have not been fully explored through the lens of discrete optimization.

In recent years, the mining industry has witnessed a marked shift towards automation and electrification of operations (Leung, Hill, & Melkumyan, 2023; Bao, Knights, Kizil, & Nehring, 2023). This trend, driven by the need for increased efficiency, safety, and environmental sustainability (Onifade et al., 2024); (Onifade, Said, & Shivute, 2023), has created new challenges in coordinating drilling operations. Our industrial partner notes that as mines adopt electric drilling machines, the complexity of managing power supply connections and optimizing drill patterns has increased significantly. These developments underscore the need for advanced optimization techniques in drilling operations.

This paper introduces the Drill Coordination Problem (DCP), a novel approach to optimizing drilling operations in open-pit mines. The DCP focuses on the assignment of target subsets to each drilling machine and the scheduling of their tasks to maximize drilling efficiency within a specified timeframe. This problem is characterized by unique constraints, including:

1. Continuous power supply connections for electrical drilling machines, limiting movement range and creating complex routing constraints.
2. Dynamic obstacles formed by drilled blast holes, which progressively alter feasible paths for machines.
3. Safety-mandated distance requirements between operating machines, introducing spatial and temporal constraints.

These unique challenges create an optimization problem that differs significantly from traditional scheduling or routing problems. Unlike typical vehicle routing problems, the DCP must account for the continuous power supply constraint and the dynamic obstacles created by drilled holes. It also differs from standard parallel machine scheduling problems due to the spatial constraints and strong interdependencies among the machines.

A critical aspect of the DCP is the potential for deadlocks. Since machines cannot switch positions, poor coordination of tasks can lead to situations where machines block each other's paths, resulting in a complete halt of operations. Such deadlocks are detrimental to productivity and can lead to significant revenue losses. Moreover, the safety distance requirement, while essential for operational safety, introduces additional complexity and can slow down the drilling process.

The significance of solving the DCP extends beyond merely increasing drilling efficiency. As our literature review reveals, while considerable research has focused on optimizing various aspects of mining operations, there is a notable gap in addressing the specific challenges of coordinating multiple electric drilling machines. By tackling this problem, we aim to contribute to the broader goal of enhancing overall mining productivity and safety in an increasingly automated industry landscape. Effective solutions to the DCP can prevent productivity-hampering deadlocks, ensure adherence to crucial safety standards, and optimize the utilization of drilling equipment.

The main contributions of this paper are:

1. A formal definition of the Drill Coordination Problem (DCP), a novel optimization problem that captures the unique constraints of coordinating electric drilling machines in open-pit mines.

2. A constraint programming formulation that efficiently models the complex spatial and temporal constraints of the DCP, including power supply limitations, dynamic obstacles, and safety requirements.
3. An effective solution approach that combines the constraint programming model with a heuristic algorithm, demonstrating the ability to solve real-world instances of the DCP.

This work represents a continuation of the ongoing trend of applying operations research techniques to mining engineering challenges (Newman, Rubio, Caro, Weintraub, & Eurek, 2010; Kozan & Liu, 2011; Chowdu, Nesbitt, Brickey, & Newman, 2022). By leveraging advanced constraint programming methods to solve a complex mining problem, we demonstrate the value of interdisciplinary approaches in addressing real-world industrial challenges.

The remainder of this paper is structured as follows: Section 2 reviews relevant literature in the field of mining optimization, with a focus on drilling operations. Section 3 provides a detailed description of the DCP. Section 4 presents our solution methodology, including the constraint programming formulation and heuristic algorithm. Section 5 discusses the results of computational experiments. Finally, Section 6 concludes the paper and suggests directions for future research.

2 Literature review

Discrete optimization has played a crucial role in enhancing the operational efficiency of mining operations by providing decision support systems for planning activities at various levels: strategic, tactical, and operational. These optimization techniques contribute to decisions ranging from the order in which mineral deposits are mined to maximize Net Present Value, to mid-term planning of equipment positioning, and day-to-day allocation of mining equipment (Newman et al., 2010; Kozan & Liu, 2011; Chowdu et al., 2022). The application of discrete optimization to these planning levels in mining has garnered significant attention from the research community due to their practical importance, with extensive literature reviews dedicated to mine design (Mwangi Akisa et al., 2021), mine production (Osanloo et al., 2008; Fathollahzadeh et al., 2021), and transportation (Alarie & Gamache, 2002; Moradi Afrapoli & Askari-Nasab, 2019).

In examining the applications of discrete optimization in the mining field, it is evident that much research has focused on improving the operational efficiency of loading and hauling. Additionally, significant work has been done in optimizing blasting operations, including the use of genetic algorithms for blasting pattern optimization (Azarafza, Feizi Derakhshi, & Jeddi, 2017), discrete-event simulation models for optimal blast design (Nageshwaraniyer, Kim, & Son, 2015), and the application of gene expression programming and grasshopper optimization algorithms to minimize blast-induced dust emission (Hosseini, Monjezi, & Bakhtavar, 2022). Recent research has also explored multi-objective optimization approaches, such as Monte Carlo simulation-based multi-objective grey wolf optimization for environmentally friendly and optimum fragmentation in mine-to-crusher planning (Hosseini, Mousavi, Monjezi, & Khandelwal, 2022).

When it comes to drilling optimization, a significant body of literature exists outside the field of discrete optimization. Many of these studies take the form of empirical investigations aimed at answering various drilling-related questions, such as identifying the best drilling equipment (Afeni, 2009), determining the optimal drilling technique (Servet et al., 2014), finding the optimal drill bit replacement time (Ugurlu & Kumral, 2020b; Uğurlu, 2021), understanding the parameters affecting the rate of penetration (Rais, Kara, Riheb, Gadri, & Khochmen, 2017; Inanloo Arabi Shad, Sereshki, Ataei, & Karamoozian, 2018; Shangxin et al., 2020; Shen, Wang, Cao, & Liu, 2022), and determining the optimal drilling parameters (Ugurlu & Kumral, 2020a). While these studies have provided valuable insights into specific aspects of drilling optimization, they generally focus on enhancing the performance of individual drills. This approach has limitations when it comes to improving the overall operational efficiency of drilling activities. In reality, drilling operations involve multiple drilling machines operating

in close proximity to each other, working as a team to complete a drilling pattern. Therefore, optimizing drilling operations requires considering the coordination and interactions between these machines as a collective effort, rather than optimizing them individually. This is where discrete optimization can play a crucial role, offering tools to address the complex interdependencies in multi-machine drilling operations.

Recent research has begun to address the optimization of drilling operations from various perspectives. Several studies have focused on optimizing the placement of additional drill holes to enhance mineral resource classification and exploration efficiency. These include the proposal of a semi-greedy optimizer for drillhole location optimization (Dutaut & Marcotte, 2020), the introduction of novel objective functions derived from Bayesian optimization to locate additional drillholes (Jafrasteh & Suárez, 2021), comparison of different geostatistical cost functions for optimizing drill hole placement (Maleki et al., 2024), a multi-objective approach for optimizing the layout of additional boreholes in mineral exploration (Hossein-Morshedy, Khorram, & Emery, 2023), and the development of a Tabu Search algorithm for positioning mining drillholes with block uncertainty (Zagré, Marcotte, Gamache, & Guibault, 2019). While these studies significantly advance our understanding of optimal drill hole placement, they primarily focus on the strategic level of drilling operations and do not address the operational challenges of coordinating multiple drilling machines in real-time operations.

Some researchers have attempted to bridge the gap between strategic planning and operational drilling by considering the coordination of drilling operations within the broader context of mine production. Notably, (Scheduling, 2016) and (Open pit mining, 2017) proposed multi-stage mine production timetabling models to optimize drilling, blasting, and excavating operations. Their approach formulates the problem as a mixed integer programming model, considering multiple stages and equipment types simultaneously. They define "mining jobs" as aggregations of several same-grade block units on the same bench, typically consisting of about 22 strategic-level blocks (each 10m x 10m x 15m). Their model aims to maximize throughput and minimize total idle times of equipment at each stage over an 18-week operational horizon. While this approach provides valuable insights into medium-term production planning and resource allocation, it operates at a coarser level of granularity than our work. For instance, their drilling stage considers total drilling meters per mining job, with drilling rates around 50 meters/hour, whereas our model focuses on individual drill hole locations. Furthermore, their approach, while comprehensive in its multi-stage view, does not address the detailed, day-to-day coordination of individual drilling machines at the blast hole level that our work targets. Despite these differences, Kozan and Liu's work represents a significant step in applying advanced optimization techniques to mine production scheduling and provides a foundation upon which our more granular, short-term planning model builds.

To date, only a few works have addressed the issue of coordinating multiple drilling machines in open-pit mines. Notable contributions include the research by (Mansouri, Andreasson, & Pecora, 2016) and (Mansouri, Lagriffoul, & Pecora, 2017), which explores the automation of diesel-powered drilling machines from a robotics perspective. These studies present valuable approaches to solving complex multi-robot planning problems in mining environments.

(Mansouri et al., 2016) propose a hybrid reasoning method for multi-robot drill planning, focusing on fleet automation that involves task allocation, motion planning, and coordination. Their approach uses heuristically-guided search to find mutually feasible solutions to sub-problems, with an emphasis on online contingency handling and domain-specific constraints. Building on this, (Mansouri et al., 2017) introduce the Multi-Vehicle Routing Problem with Dense Dynamic Obstacles (MVRP-DDO), which accounts for nonholonomic constraints and dynamic obstacles. They propose a multi-abstraction search approach to compute executable plans for multiple drilling machines in highly constrained environments.

While these works provide significant insights into the coordination of drilling machines, our research complements these contributions in several key aspects. Firstly, we focus specifically on electrical drilling machines, which must maintain a continuous connection to their power source. This introduces

unique constraints not present in diesel-powered machine coordination. Secondly, while previous works utilize heuristic methods and multi-abstraction search, we propose a constraint programming formulation, offering a different perspective on solving the Drill Coordination Problem (DCP). Our approach explicitly considers the constraints imposed by power cables, adding a layer of complexity to the spatial and temporal constraints of the problem. Furthermore, we focus on maximizing drilling efficiency within a specified time frame, considering the interplay between machine assignments, scheduling, and cable-related constraints.

Recent research has made significant strides in optimizing various aspects of drilling operations, from strategic placement to multi-robot coordination. However, a critical gap remains in addressing the unique challenges posed by the increasing adoption of electric drilling machines in open-pit mining. Although electric vehicles are not new to the mining industry, their prevalence is expected to grow substantially, with projections indicating that nearly 50% of mining vehicle sales will be electric vehicles by 2044, valued at over US$23 billion (Jaswani & Jeffs, 2024). This trend towards electrification introduces new dimensions to existing optimization problems, particularly in terms of power supply management and machine coordination. These challenges, combined with the complex spatial and temporal constraints of drilling operations, create a novel optimization problem that differs significantly from traditional scheduling or routing problems.

The Drill Coordination Problem (DCP) introduced in this paper addresses this gap by considering the coordination and interactions between multiple electric drilling machines as a collective effort. It offers a new perspective on drilling optimization that focuses on maximizing drilling efficiency while considering the interplay between machine assignments, scheduling, and cable-related constraints. By proposing a constraint programming formulation, our approach provides a fresh perspective on solving the DCP, explicitly accounting for the unique constraints imposed by power cables and the need for continuous power connection.

3 Problem description

The Drill Coordination Problem (DCP) in open-pit mining presents a complex challenge that requires careful consideration of multiple operational constraints and optimization objectives. To provide a comprehensive understanding of this problem, we approach its description from three complementary perspectives. First, we offer a conceptual overview that explains the key elements and constraints of the problem. Next, we present an illustrative example to demonstrate how these concepts apply in a practical scenario. Finally, we provide a mathematical formulation that precisely defines the problem's structure and constraints. This multi-faceted description aims to give readers a thorough grasp of the DCP, laying the groundwork for the solution approaches discussed in subsequent sections.

3.1 Conceptual overview

Figure 1 provides an illustrative example of the Drill Coordination Problem (DCP), showcasing a set of targets denoted as $\mathcal{J}$ and a set of drilling machines denoted as $\mathcal{M}$, each connected to its power source. This representation is used to explain the key concepts and constraints of the problem as defined by our industrial partner and the physical limitations of the drilling equipment.

In this illustration, each target is identified by its number and its corresponding row and column position. The first row (row 1) and the first column (column 1) contain target 1. While this example shows 45 targets and 3 machines, real-world instances may involve different numbers of targets and machines depending on the specific mining operation and bench size.

While our model focuses on the scheduling aspect, it's important to understand how operational specifications relate to our model's parameters. Blast hole characteristics such as diameter, depth, burden, and spacing, along with drilling parameters like rotational speed and thrust force, all influence the model's key parameters:

- Drilling time (p_{mj}): Encapsulates factors like hole depth, rock properties, and machine capabilities.
- Setup time (s_{ij}^m): Reflects the time for machine movement, influenced by burden and spacing.
- Safety distance (d): Determined by burden and spacing, equipment size, cable management, and safety standards.

These operational aspects are implicitly represented in our model parameters, allowing us to focus on the scheduling optimization while accounting for the underlying physical constraints.

All targets in this example share the same characteristics, including identical blast hole diameter and depth. However, the drilling time at each target may vary depending on the machine performing the drilling and the local rock properties. Each machine is capable of drilling at any target. Once a machine begins drilling, the operation cannot be interrupted. Additionally, once a blast hole is drilled, neither the machine nor its cable can pass over it.

When a machine moves from one target to another, its cable adjusts by either extending or retracting. As per the requirements of our industrial partner and the constraints imposed by the power cables, the machines move between columns in the same direction, from left to right. This means that a machine must first visit the targets assigned to it in the leftmost column before progressing towards those located on the right. Furthermore, the machines must maintain a safe distance from each other while operating, as mandated by safety regulations.

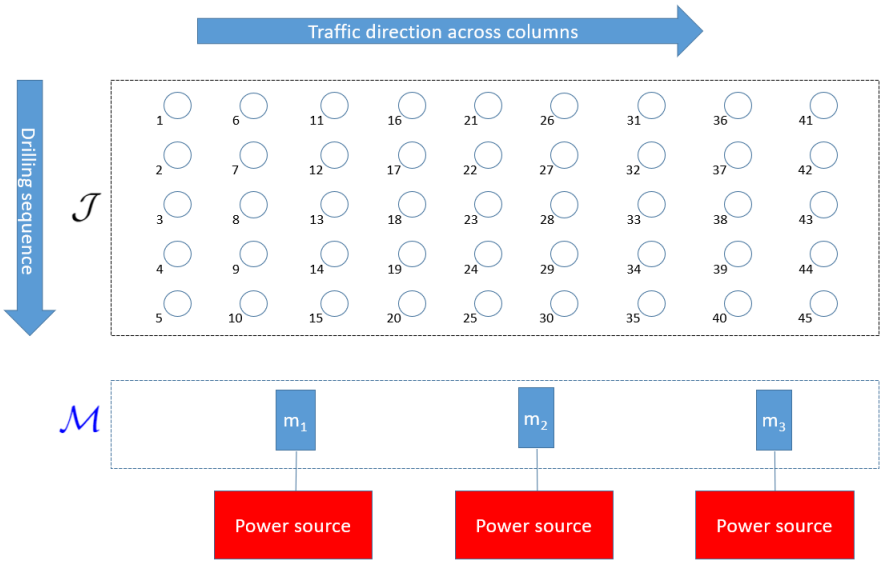


Figure 1: An illustrative example of the DCP

Let us examine these rules in detail:

- The drilling machines must remain connected to their power sources, requiring them to maintain the same relative positions throughout the drilling process. If machine m is initially positioned to the left of machine n, they must stay in that order during their operation. Otherwise, collisions may occur between the machines or between a machine and its neighboring cable.
- Due to the inability of both machines and cables to traverse over blast holes, tasks assigned to them within a column must be performed in reverse order, starting from the task furthest from the column entrance and progressing towards it. For instance, if there was only one machine performing the drilling pattern depicted in Figure 1, it would follow the numerical order of the targets $(1, 2, 3, \ldots, 45)$.

- To prevent premature wear of the power cables and avoid burying previously drilled holes, machines are prohibited from moving directly to targets in different columns, as it would drag the cable along the ground. Instead, a machine must exit its current column (again, by moving backwards) before entering the next column through its entrance to access a target in a different column.
- The movement restrictions within and between columns give rise to precedence constraints over targets within a column. Targets belonging to the same column must be visited in their numerical order as depicted in Figure 1, even if they are assigned to different machines. For example, in column 3, the targets have to be visited in the order 11, 12, 13, 14, 15, because, if, for example, target 12 were to be visited before target 11, this would render target 11 impossible to access. Indeed, given that a machine cannot move over a blast hole, target 11 would be inaccessible via column 3. And since the machines cannot move directly between columns, target 11 would also be inaccessible via all other columns. This constraint differs from typical precedence constraints found in other combinatorial optimization problems, which usually only consider precedence within the same resource assignment. These precedence constraints resemble those described in (van den Akker, Hoogeveen, & van Kempen, 2006) (Section 6).
- Another constraint involves maintaining a minimum distance between machines and the cables of neighboring machines, measured in terms of columns. For example, if the safety distance is set to 1 column, each pair of machines must have at least one column between them. Consequently, when a machine starts drilling at a target, the neighboring targets become temporarily inaccessible. Targets are considered neighbors if the number of columns separating them is less than the minimum safety distance. Figure 2 illustrates this concept, where machine 1 drilling at target 13 renders the neighboring targets (colored in grey) inaccessible.

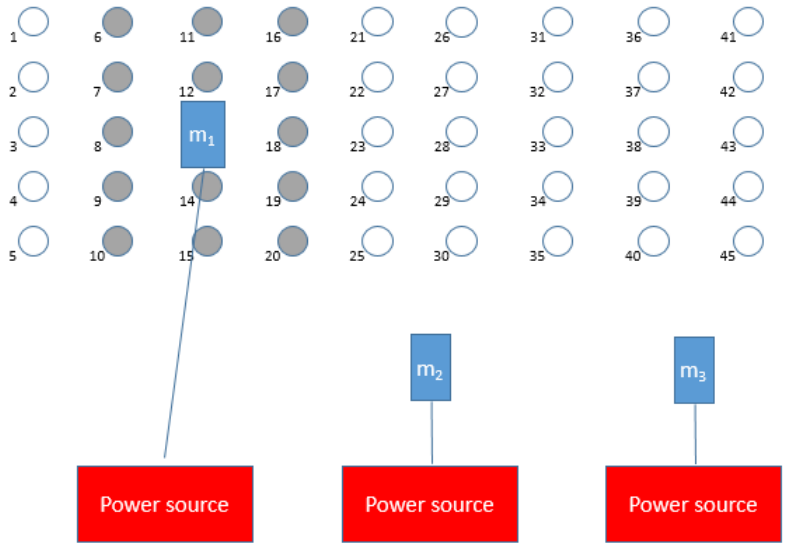


Figure 2: Target 13's neighbours

Given these operational constraints and the complex interdependencies between drilling machines, our model aims to solve the Drill Coordination Problem (DCP) with the following specific objectives :

1. Maximize the number of holes drilled within a given time horizon.
2. Determine the optimal assignment of tasks (blast holes) to drilling machines.
3. Generate a feasible schedule for each machine that respects all operational constraints, including power cable management, safety distances, and precedence requirements.

By addressing these objectives, our model seeks to optimize the overall efficiency of the drilling operation, balancing the workload across machines while adhering to the complex set of constraints inherent in open-pit mining operations. This optimization has the potential to significantly improve productivity, reduce operational costs, and enhance safety in mining operations.

Note 1. In real-world situations, targets seldom align in perfect rows and columns. The target layouts depicted in Figure 1 and throughout this paper are for explanatory purposes only. Even with irregular target arrangements, effective coordination is achievable through column-based grouping. The models presented in this paper are designed to handle messy target layouts, not just grid-like ones. The essential factor is the clustering of targets into columns, regardless of whether the columns are perfectly straight, evenly spaced, or have an equal number of targets.

3.2 Illustrative example

Before delving into the mathematical and constraint programming formulations of the problem, let us examine a small instance to illustrate the aforementioned points. To streamline computations and focus on the critical aspects of the problem, we will assume that each machine operates at a constant speed when performing its assigned tasks. Machine 1 can complete any blast hole drilling in 2 minutes, machine 2 takes 4 minutes, and machine 3 requires 3 minutes. We will also assume a safety distance of one column, and that there is no time required to move between targets. The optimal solution, depicted in Figure 3, showcases the assigned machines through color-coding of the targets, and the start time of each operation is specified within the corresponding circles representing the targets. Additionally, it is important to note that the planning horizon for this example is 20 minutes.

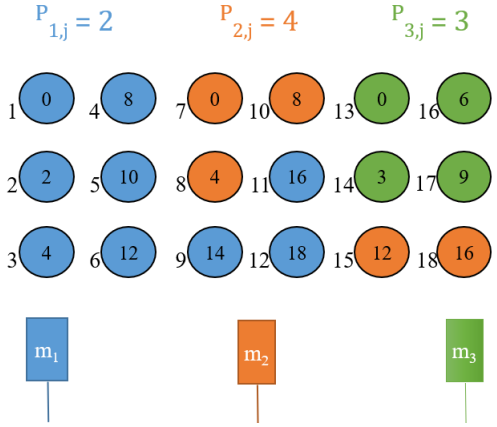


Figure 3: A solved instance of the DCP

To gain a better understanding of the solution, let us examine the path followed by machine 1. It starts at target 1 at time 0 and sequentially performs all the tasks in column 1 without any interruption, finishing at time 6 (4 + 2). After completing its tasks in column 1, machine 1 has to wait for two time units before accessing column 2. This delay is due to machine 2 drilling at target 8 in column 3, which is scheduled to finish at time 8 (4 + 4). Once machine 2 completes its final task in column 3, it proceeds to column 4. At this point, machine 1 is allowed to move to column 2 without violating the spacing constraint. Machine 1 then continues its operations in columns 2, 3, and 4 without any interruptions, ultimately finishing at time 20. It's important to note that throughout the process, the machines maintain their relative positions, meaning that a machine never finds itself to the right of another machine that was initially positioned to its right. Additionally, the machines move from column to column in a rightward direction and within each column, they move downward.

3.3 Mathematical description

While our primary solution approach uses constraint programming (CP), we present here a mixed-integer programming (MIP) formulation to provide a precise mathematical description of the problem. This formulation serves to clarify the problem's structure and constraints for readers familiar with mathematical programming. However, as we will show, this MIP model is not suitable for solving large-scale, real-world instances due to computational limitations. Our CP model, presented in Section 4.1, is more efficient for practical problem sizes.

As discussed in the preceding sections, the coordination of drills primarily involves assigning targets to drilling machines and determining the start time for drilling at each target. Since the problem does not involve a routing aspect, we will treat it as a scheduling problem. Consequently, we will employ scheduling terminology to describe the problem in the subsequent sections. We will refer to targets as tasks/jobs, the time required to move between targets as setup time, the time needed to dig a blast hole as processing time, and so on.

Our proposed mixed-integer program is based on a model for parallel machine scheduling problems with time-indexed variables x_{mjt}. Time is represented by a discrete set $\mathcal{T} = \{0, 1, 2, \ldots, H - p^*\}$, where H is the planning horizon and p^* is the shortest processing time. Each element of the set $\mathcal{T}$ represents a moment at which the execution of a task can begin. For example, if a machine m starts processing a job j at time t, and the process lasts for p_{mj} time units, the machine will be occupied during the interval $[t, t + p_{mj}[$ and will be free again starting from $t + p_{mj}$. As for the restrictions on the directions of movement, they will be imposed through the travel time matrix. Travel times between two positions in a direction that violates these restrictions will have prohibitive values.

It is worth noting that the primary purpose of this mixed-integer program is to provide a comprehensive mathematical representation of the problem for a broad optimization community audience. Despite its theoretical appeal, our experiments showed that commercial solvers could only tackle small toy instances, like the one highlighted in the prior section, and struggled with practical, real-world-sized instances. Hence, its practical applicability is limited. As we will demonstrate in subsequent sections, our constraint programming (CP) model proves much more efficient and suitable for solving practical instances of this problem.

The proposed model uses the following notation:

Sets	
$\mathcal{M}$	the set of machines, indexed by m and n.
$\mathcal{J}$	the set of jobs, indexed by i and j.
$\mathcal{I}_j$	the set of tasks that are inaccessible when task j has begun.
$\mathcal{C}$	the set of columns, indexed by c.
$\mathcal{C}^+$	the set of columns, to which we add dummy columns (on both sides), indexed by c (see note 2 below).
$\mathcal{J}_c$	the set of tasks that belong to column c, indexed by j.
$\mathcal{T}$	the set of time periods, indexed by t.
Parameters	
n	the number of jobs per column.
p_{mj}	the processing time of task j on machine m.
s_{ij}^m	the setup time between tasks i and j.
d	the number of columns that have to be kept empty between pairs of machines.
H	the planning horizon.

Note 2. The setup time s_{ij}^m is finite if j is the successor of i or if the column to which j belongs comes after the column to which i belongs. The travel time s_{ij}^m is infinite in all other cases. This ensures that machines adhere to the directions of movement without the need for explicit constraints in the model.

Variables

$x_{mjt} = \begin{cases} 1, \text{if the machine } m \text{ begins the job } j \text{ at time } t; \\ 0, \text{otherwise.} \end{cases}$

$y_{mct} = \begin{cases} 1, \text{if the machine } m \text{ is active in the column } c \in \mathcal{C}^+ \text{ at time } t; \\ 0, \text{otherwise.} \end{cases}$

C_j the completion time of job j.

Note 3. The inclusion of dummy columns allows the solver to assign columns to machines that fall outside of the pattern when they are inactive. Understanding the necessity of these columns requires examining the consequences if they were absent. Let us consider the instance depicted in Figure 3 as an illustrative example. In this scenario, no job to the right of machine 3 can be executed while it is active, as it would result in collisions. However, even when machine 3 is inactive, the solver still needs to assign a column to it (denoted by variable y_{3ct}) within the restricted index set $\mathcal{C}$. This limitation prevents machine 3 from deviating from the established pattern, confining it to the last column. This constraint poses a problem, as the proximity constraint will render jobs close to machine 3 inaccessible to other machines. Consequently, a subset of jobs can only be accessed by the rightmost machine, potentially compromising the optimal solution.

Model 1

Maximize
$$Z = \sum_{m\in\mathcal{M}} \sum_{j\in\mathcal{J}} \sum_{t\in\mathcal{T}} x_{mjt} \tag{1.1}$$

s.t.
$$\sum_{m\in\mathcal{M}} \sum_{t=0}^{H-p_{mj}} x_{mjt} \leq 1, \qquad \forall j \in \mathcal{J}, \tag{1.2}$$

$$x_{mjt} + \sum_{h=max\{0,\ t-s^m_{ij}-p_{mi}+1\}}^{\min\{H-p_{mi},\ t\}} x_{mih} \leq 1, \qquad \forall i,\ j \in \mathcal{J},\ i \neq j,\ \forall m \in \mathcal{M},\ \forall t = 0,\dots,H-p_{mj}, \tag{1.3}$$

$$\sum_{m\in\mathcal{M}} \sum_{j\in \bigcup_{k=c}^{c+d} \mathcal{J}_k} \sum_{h=max\{0,\ t-p_{mj}\}}^{\min\{H-p_{mj},\ t\}} x_{mjh} \leq 1, \qquad \forall c \in \{1,\dots,|\mathcal{C}|-l\},\ \forall t \in \mathcal{T}, \tag{1.4}$$

$$\sum_{j\in\mathcal{J}_c} \sum_{h=max\{0,\ t-p_{mj}\}}^{t} x_{mjh} \leq y_{mct}, \qquad \forall t \in \mathcal{T},\ \forall m \in \mathcal{M},\ \forall c \in \mathcal{C}, \tag{1.5}$$

$$\sum_{c\in\mathcal{C}^+} y_{mct} \leq 1, \qquad \forall t \in \mathcal{T},\ \forall m \in \mathcal{M}, \tag{1.6}$$

$$\sum_{c\in\mathcal{C}^+} c\ y_{nct} - d - 1 \geq \sum_{c\in\mathcal{C}^+} c\ y_{mct}, \qquad \forall t \in \mathcal{T},\ \forall m,\ n \in \mathcal{M},\ n \prec m, \tag{1.7}$$

$$\sum_{m\in\mathcal{M}} \sum_{t=0}^{H-p_{mj}} x_{mjt}(t+p_{mj}) = C_j, \qquad \forall j \in \mathcal{J}, \tag{1.8}$$

$$\sum_{m\in\mathcal{M}} \sum_{t=0}^{H-p_{mj}} t\ x_{mjt} \geq C_i, \qquad \forall i,\ j \in \mathcal{J},\ i \prec j, \tag{1.9}$$

$$x_{mjt} \in \{0,1\}, \qquad \forall j \in \mathcal{J},\ \forall m \in \mathcal{M},\ \forall t = 1,\dots,H-p_{mj}, \tag{1.10}$$

$$y_{mct} \in \{0,1\}, \qquad \forall t \in \mathcal{T},\ \forall m \in \mathcal{M},\ \forall c \in \mathcal{C}^+, \tag{1.11}$$

$$C_j \geq 0, \qquad \forall j \in \mathcal{J}. \tag{1.12}$$

The objective function (1.1) maximizes the total number of jobs started within the time horizon. Note that in our formulation, starting a job is equivalent to completing it, as we do not allow for job interruptions. Therefore, maximizing the number of started jobs effectively maximizes the number of completed jobs.

The formulation (1.1)–(1.3) represents a parallel machine scheduling problem aimed at maximizing the total number of completed jobs within the specified time horizon.

Constraint group (1.4) ensures the proper spacing between machines by allowing only one machine to be active when any job within the same neighborhood (from columns c to $c+l$) is being executed.

Constraint group (1.5) initializes the variables y_{mct}, indicating the activity of machine m in column c at time t, when it initiates job j within the time interval $[t - p_{mj} + 1, t]$.

Constraint group (1.6) prevents a machine from occupying multiple positions simultaneously.

Constraint group (1.7) guarantees collision-free operation between machines. It mandates that the column of machine m be strictly less than the column of machine n for all pairs where n precedes m, ensuring machine m always remains to the left of machine n. By imposing the condition that the column of machine m must be less than the column of machine n minus $d - 1$, the spacing constraint is enforced, rendering constraint group (1.6) redundant.

Constraint group (1.8) initializes the variables C_j.

Constraint group (1.9) ensures that a machine does not pass over a previously drilled position (i.e., a hole) by requiring the completion time of job i to be less than or equal to the start time of job j for all pairs where i precedes j.

Constraint groups (1.10)–(1.12) define the domains of the decision variables.

To assess the model's correctness, we implemented it using the CPLEX commercial solver with default parameters. Our tests confirmed that the model produces feasible solutions that respect all problem constraints. However, these tests also revealed the model's limitations in terms of problem size: it was capable of solving only very small instances, with a maximum of 18 jobs and 3 machines, within reasonable time limits. This limitation underscores the need for more efficient solution approaches, such as the constraint programming model we present in Section 4.1, for addressing real-world problem instances.

4 Solution methodology

In this section, we present our approach to solving the Drill Coordination Problem (DCP). We propose two complementary methods: a constraint programming (CP) formulation and a heuristic algorithm. The CP formulation provides an exact solution method capable of handling real-world problem instances, while the heuristic algorithm offers a quick way to generate initial solutions and serves as a benchmark for evaluating the CP model's performance.

4.1 Constraint programming formulation

We present a constraint programming model of the problem, utilizing an extension of constraint programming specifically designed for scheduling problems. In this modeling framework, decision variables are represented as interval variables, unlike classical constraint programming where integers denote task starting times. Interval variables are well-suited for our purposes as they encapsulate various task characteristics (such as starting time, ending time, and processing time) within a single variable.

The interval variables can be optional, meaning not all variables require assignment in the final solution. Optional variables are employed to represent potential task assignments to machines. Additionally, we utilize interval sequence variables to represent the set of tasks assigned to each machine. State functions are another essential component of our model. They enable us to track the state of a machine while it performs a task, primarily monitoring the machine's positions during drilling operations. For a comprehensive overview of this modeling framework, we refer the reader to (Laborie, Rogerie, Shaw, & Vilím, 2018).

The proposed model uses the following notation:

Sets	
$\mathcal{M}$	the set of machines, indexed by $m \in \{1,\ 2,\ \ldots,\ |\mathcal{M}|\}$.
$\mathcal{M}^*$	$\{2,\ \ldots,\ |\mathcal{M}|\}$, the set of machines deprived from its first element.
$\mathcal{J}$	the set of tasks, indexed by i and $j \in \{1,\ 2,\ \ldots,\ |\mathcal{J}|\}$.
$\mathcal{I}_j$	the set of tasks that are inaccessible when task j has begun.
$\mathcal{C}$	the set of columns, indexed by $c \in \{1,\ 2,\ \ldots,\ |\mathcal{C}|\}$.
$\mathcal{J}_c$	the set of tasks that belong to column c, indexed by j.
$\mathcal{J}_c^-$	the set $\mathcal{J}_c$ deprived of its last element.
Parameters	
col_j	the column to which task j belongs.
p_{mj}	the processing time of task j on machine m.
s_{ij}^m	the setup time between tasks i and j.
d	the number of columns that have to be kept empty between pairs of machines.
H	the time horizon.
Variables	
x_{jm}	optional interval variable of size p_{jm} indicating if task j is performed by machine m.
y_j	optional interval variable that represents a task j.
z_m	sequence of interval variables x_{jm} on machine m.
l_m	state function that represents the location of machine m.

Note 4. When we omit the index of a variable or parameter inside the model, it indicates that we are referring to the array of variables or parameters encompassing the omitted index. For instance, $\mathbf{x}_{j\cdot}$ represents the one-dimensional array $[x_{j1},\ x_{j2}, \ldots,\ x_{j|\mathcal{M}|}]$ while $\mathbf{s}_{\cdot\cdot}^m$ represents the two-dimensional array of setup times between each pair of tasks for a given machine m.

Model 2

$$\text{Maximize} \qquad Z = \sum_{m \in \mathcal{M}} \sum_{j \in \mathcal{J}} \texttt{PresenceOf}(x_{jm}) \tag{2.1}$$

$$\text{s.t.} \qquad \texttt{Alternative}(y_j,\ \mathbf{x}_{j\cdot}), \quad \forall j \in \mathcal{J}, \tag{2.2}$$

$$\texttt{NoOverlap}(z_m,\ \mathbf{s}_{\cdot\cdot}^m), \quad \forall m \in \mathcal{M}, \tag{2.3}$$

$$\texttt{EndBeforeStart}(y_j,\ y_{j+1}), \quad \forall c \in \mathcal{C},\ \forall j \in \mathcal{J}_c^-, \tag{2.4}$$

$$\texttt{NoOverlap}(\{y_j \mid j \in \mathcal{I}_j\}), \quad \forall j \in \mathcal{J}, \tag{2.5}$$

$$\texttt{PresenceOf}(y_j) \implies \texttt{PresenceOf}(y_{j-1}), \quad \forall j \in \mathcal{J} \setminus \{1\}, \tag{2.6}$$

$$\texttt{AlwaysEqual}(l_m,\ x_{jm},\ col_j), \quad \forall j \in \mathcal{J},\ \forall m \in \mathcal{M}, \tag{2.7}$$

$$\texttt{AlwaysIn}(l_m,\ x_{j,m-1},\ col_j + d + 1,\ |\mathcal{C}| + d + 1), \quad \forall j \in \mathcal{J},\ \forall m \in \mathcal{M}^*. \tag{2.8}$$

The objective (2.1) aims to maximize the number of scheduled tasks within the planning horizon.

The constraint set `Alternative`(*interval_var, list_of_intervals*) (2.2) takes an optional interval variable (y_j) and a list of optional interval variables ($x_{j\cdot}$) as arguments. It ensures that only one optional variable from the list will be present in the solution if the optional variable y_j is present. This constraint also enforces that the interval variable y_j has the same start and end time as the selected optional variable x_{jm}. Thus, this constraint assigns each task to a specific machine.

The constraint set `NoOverlap`(*list_of_intervals, array_of_setup_times*) (2.3) ensures, as the name suggests, that tasks assigned to the same machine do not overlap. Additionally, it ensures that the elapsed time between any pair of scheduled tasks is greater than the corresponding setup time.

The constraint set `EndBeforeStart`(*interval_var1, interval_var2*) (2.4) represents a precedence constraint, ensuring that the task represented by *interval_var1* ends before the task represented by *interval_var2* can begin.

The constraint set `NoOverlap`(*list_of_intervals*) (2.5) guarantees that neighboring tasks are not performed simultaneously. The concept of neighboring tasks was defined in the previous section.

The constraint set (2.6) ensures task sequence integrity within the drilling schedule. Specifically, it enforces that if a task j is scheduled, then its immediate predecessor task $j-1$ must also be scheduled.

This maintains the continuity of operations, guaranteeing that no tasks are skipped in the sequence of scheduled blast holes. This constraint is crucial for preserving the logical order of tasks, as it ensures that the drilling operations follow a consecutive pattern, reflecting the physical requirement that a machine cannot bypass any preceding targets in the same column.

The constraint set `AlwaysEqual`(*state_variable, interval_var, value*) (2.7) fixes the state/position of machine m to the value col_j, indicating the column to which task j belongs, while machine m is performing task j.

The constraint set `AlwaysIn`(*state_variable, interval_var, min, max*) (2.8) restricts the value that p_m can take to the interval $[min, max]$, where min represents the previous machine's column plus the safety distance, and max represents the furthest column in which machine m can be located. This constraint is applicable only while machine $m-1$ is operating.

Note 5. Similarily to the MIP model, the restrictions on the movement of machines between columns are enforced through the setup times array. Prohibitive setup times are inserted between pairs of targets that should not be visited consecutively.

Note 6. The parameter H is not explicitly included in the model because it is incorporated within the definition of the x_{jm} variables. These variables are defined with a task completion time within the range $[0, H]$.

Note 7. The `AlwaysEqual` constraint in (2.7) includes additional alignment parameters:

- *isStartAligned*: When true, it ensures that the start of x_{jm} coincides with the start of an interval in the state function l_m.
- *isEndAligned*: When true, it ensures that the end of x_{jm} coincides with the end of an interval in the state function l_m.

These alignment parameters are important for precise synchronization between the task intervals and the state function intervals. In this model, both parameters are set to *True*.

4.2 Heuristic algorithm

While our CP formulation provides an exact solution method, we have also developed a heuristic algorithm for the DCP. This heuristic serves two purposes:

1. It provides a method for quickly generating feasible solutions, which can be used as initial solutions for the CP model to potentially accelerate the solving process.
2. It serves as a benchmark against which we can compare the performance of our CP model, given the lack of existing algorithms specifically designed for this problem.

The heuristic operates in two main phases: task assignment and scheduling. Here, we provide a brief description of each phase before presenting the algorithm in detail and then we illustrate its application with a small example.

Assignment

Given a set of machines $\mathcal{M} = 1, \ldots, M$, a set of tasks $\mathcal{J} = 1, \ldots, N$, a set of columns $\mathcal{C} = 1, \ldots, C$, and the processing time p_{mj} of task $j \in \mathcal{J}$ on machine $m \in \mathcal{M}$, our heuristic assigns a subset of consecutive columns to each machine.

In this assignment, machine 1 is assigned the tasks from columns 1 to c_1 (the first c_1 columns), machine 2 is assigned the tasks from columns c_1+1 to c_1+c_2, and so on, until machine M is assigned the tasks from columns $\sum_{m=1}^{M-1} c_m + 1$ to C. The quantity of columns, c_m, allocated to machine m is proportionate to that assigned to any other machine n by a factor α_{mn}. This proportionality factor, α_{mn}, indicates the relative speed of machine m to machine n. It is defined by the formula: $\alpha_{mn} = \frac{\tau_n}{\tau_m}$, where τ_m is the total time required for machine m to complete all tasks independently.

For example, let us consider a scenario with three machines ($M = 3$). If machine 1 operates at twice the speed of machine 2 ($\alpha_{12} = 2$) and thrice the speed of machine 3 ($\alpha_{13} = 3$), then machine 1 is assigned columns double that of machine 2 ($c_1 = \alpha_{12} \cdot c_2$) and triple that of machine 3 ($c_1 = \alpha_{13} \cdot c_3$). Machine 2 is allocated 1.5 times more columns than machine 3. To determine the unknowns c_1, c_2, and c_3, we solve the system of equations: $c_1 = \alpha_{12} \cdot c_2$, $c_1 = \alpha_{13} \cdot c_3$, and $c_1 + c_2 + c_3 = C$.

Scheduling

To determine the starting time of tasks, we initiate the process by assigning a priority level to each machine. A machine m is considered to have higher priority than another machine n if $\alpha_{mn} > 1$, indicating that machine m is faster than machine n. In case of a tie, where $\alpha_{mn} = 1$ for a pair of machines m and n, we resolve it lexicographically.

Once the machines are prioritized, we proceed to schedule the tasks assigned to the machine with the highest priority by assigning them the earliest start times within the planning horizon H. For instance, the first task j_1 on the machine with the highest priority m_1 should commence at time $t = 0$. Subsequently, the second task j_2 on the same machine should start at time $t = p_{m_1 j_1} + s^{m_1}_{j_1 j_2}$, and so on, ensuring that it does not exceed the planning horizon H. We follow this procedure for the remaining machines. By doing so, the start of a task can only be delayed by the distanciation constraints and is constrained within the planning horizon H.

The algorithm can be summarized as follows:

1. Compute $\tau_m = \sum_{j \in \mathcal{J}} p_{mj}$, $\forall m \in \mathcal{M}$
2. Compute the coefficient $\alpha_{1m} = \frac{\tau_m}{\tau_1}$, $\forall m \in 2, \ldots, M$.
3. Solve the system of equations $c_1 = \alpha_{1m} c_m$, $\forall m \in 2, \ldots, M$ and $\sum_{m \in \mathcal{M}} c_m = C$ (this can be done by a simple substitution).
4. Round to the nearest integer the value of the variables c_m, $m = 1, \ldots, M-1$ and put $c_M = C - \sum_{m=1}^{M-1} \lfloor c_m \rceil$.
5. Assign the tasks of the c_1 first columns to machine 1, the tasks of the c_2 next columns to machine 2, and continue up to the last machine.
6. Assign priority levels to the machines based on their speed (the machine $m \in \mathcal{M}$ with the smallest τ_m should have highest priority) and break ties between machines using lexicographical order.
7. Determine the starting time of the tasks by starting with the tasks assigned to the machine with the highest priority and work down the list of machines, ensuring that task start times do not exceed the planning horizon H.

Note 8. The starting time of a task j on a machine m is determined as $max(\{e_i \mid i \in \mathcal{N}_j \bigcup \{pred(j)\}\}) + s^m_{pred(j),j}$ where e_i represents the completion time of task i, $\mathcal{N}_j$ denotes the set tasks that are neighbours of task j, and $pred(j)$ refers to the task that precedes task j on the same machine.

Note 9. An alternative approach to the heuristic was tested, wherein consecutive tasks were assigned to machines, rather than assigning consecutive columns of tasks. However, this modification did not yield any improvement in the results.

To facilitate comprehension of the heuristic algorithm, let us walk through its application on a small instance of the drill coordination problem (DCP). For consistency, we will use the instance detailed in Section 3, which spans a planning horizon of 20 minutes. The problem instance is depicted in Figure 4.

Initialization

Consider the following parameters:

- The set of machines $\mathcal{M} = \{1,\ 2,\ 3\}$

- The set of tasks $\mathcal{J} = \{1, \ldots, 18\}$
- The set of columns $\mathcal{C} = \{1, \ldots, 6\}$
- Processing times as shown in Figure 4

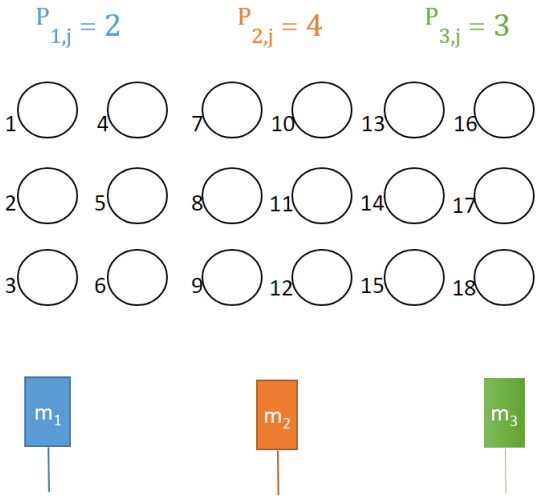


Figure 4: A small instance of the DCP

Assignment phase

Compute τ_m, the time machine m would take to complete all tasks independently. We obtain:

- $\tau_1 = 36$
- $\tau_2 = 72$
- $\tau_3 = 54$

Using τ_m, we determine the coefficients α_{m1} for $m \in \{2,\ 3\}$:

- $\alpha_{12} = 2$
- $\alpha_{13} = 1.5$

With the α_{m1} coefficients at hand, we solve the following equations to find the number of consecutive columns, c_m, for each machine m:

$$c_1 = \alpha_{12} \cdot c_2$$
$$c_1 = \alpha_{13} \cdot c_3$$
$$c_1 + c_2 + c_3 = 6$$

This results in:

- $c_1 \approx 2.8$
- $c_2 \approx 1.4$
- $c_3 \approx 1.8$

Upon rounding to integer values, we have:

- $c_1 = 3$
- $c_2 = 1$
- $c_3 = 2$

Following column assignment, Figure 5 presents the partial solution post assignment.

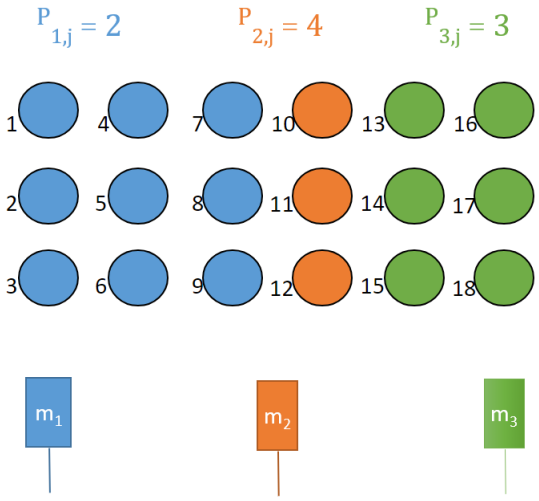


Figure 5: DCP post-assignment phase

Scheduling phase

The first step is to assign priority based on machine speed, with the machine having the smallest τ_m as the highest priority. Here, machine 1 is top priority, followed by machine 3, then machine 2.

The next step is to schedule tasks for each machine, starting with the highest priority. For machine 1, the task scheduling is straightforward as depicted in Figure 6.

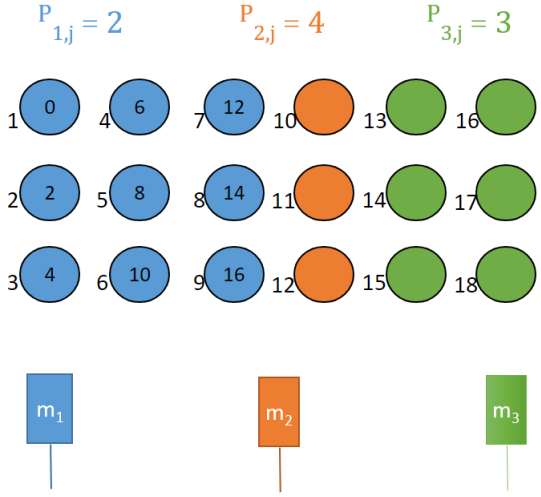


Figure 6: DCP post-scheduling for machine 1

Next on the priority list is machine 3. Given that only a single column must remain unoccupied between any two machines, and considering that machine 2's tasks are still pending, there is no potential for overlap between tasks of machines 3 and 2. As a result, scheduling tasks for machine 3 is as uncomplicated as it was for machine 1. The updated solution is shown in Figure 7.

For machine 2, task 10 can commence earliest at $t = 18$ due to dependencies on machines 1 and 2. Notably, with machine 3 occupying column 5 from $t = 0$ to $t = 9$, machine 2 is precluded from initiating Task 10 during this period. Starting task 10 at $t = 9$ isn't feasible for machine 2 either, as it would be drilling until $t = 13$ (9+4), creating a conflict with machine 1, which is drilling in column 3 from $t = 12$ to $t = 18$. Consequently, by the time machine 2 could potentially start task 10 at $t = 18$, it would overrun the 20-minute drilling window, making the scheduling of any task for machine 2 unfeasible. The final arrangement is depicted in Figure 8.

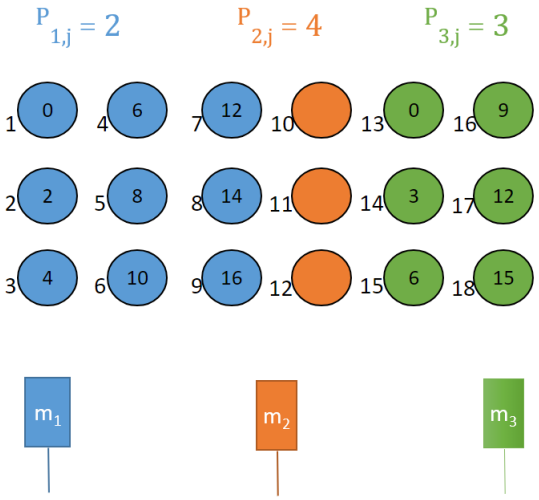


Figure 7: DCP post-scheduling for machines 1 and 3

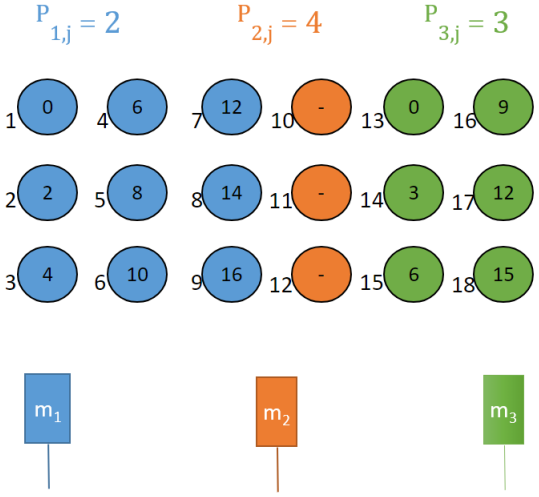


Figure 8: Complete DCP solution

In conclusion, while the heuristic approach scheduled 15 out of the 18 tasks within the given planning horizon, the optimal solution managed to schedule all 18 tasks within the same 20-minute timeframe, indicating an optimality gap of 3 tasks.

It is important to clarify that the solution provided by this heuristic algorithm does not represent a final operational plan, but rather an initial scheduling attempt within a given time horizon. In the example provided, where 15 out of 18 tasks were scheduled within the 20-minute timeframe, this does not imply that the remaining 3 tasks are abandoned or that the mine operation is left incomplete.

In practical mining operations:

- Unscheduled tasks are not discarded but are typically carried over to the next scheduling period.
- The optimization goal is to maximize efficiency within each scheduling window, not to abandon necessary work.
- Multiple scheduling periods would be used to ensure all required drilling is completed.
- In real-world scenarios, mining operations continue until all planned excavation is complete, adjusting schedules as needed.

This heuristic solution demonstrates how tasks might be initially allocated within a single time window. In practice, mining planners would use this as a starting point, making further adjustments to ensure all necessary work is completed over multiple scheduling periods if needed.

4.3 Integration of CP and heuristic approaches

In our computational experiments, we utilize both the CP formulation and the heuristic algorithm. The CP model is our primary solution method, capable of finding optimal solutions for real-world instances of the DCP. The heuristic algorithm, while not guaranteed to find optimal solutions, provides a basis for comparison and helps us evaluate the effectiveness of the CP approach.

In some instances, we use the solution generated by the heuristic as a starting point for the CP solver. However, our experiments show that this does not consistently accelerate the solving process. Nonetheless, the heuristic remains valuable as a benchmark and as a method for quickly generating feasible solutions when optimality is not critical.

The following section on computational experiments will detail how these two approaches perform on various problem instances and how they compare to each other.

5 Computational experiments

In this section, we present our computational experiments for the CP model. The experiments were performed on a laptop computer equipped with an eleventh-generation 64-bit Intel processor (i7-11800H) running at 2.30 GHz. The model was implemented using CPLEX's Python API (docplex) and solved using IBM CPLEX 22.1.0 with the default parameters. Notably, a runtime limit of 2 minutes was set for the solver in each experiment. The rationale behind this specific runtime limit is rooted in our prior observations during preliminary testing on a computer cluster. Despite extensive computation time exceeding 24 hours, the solver failed to yield optimal solutions, and further runtime did not significantly improve solution quality. Recognizing the practical constraints of real-time optimization and the need for timely responses in the face of parameter uncertainty, we adopted the 2-minute runtime limit as it allowed us to obtain good quality solutions within a short timeframe. This choice aligns with our objective of enabling re-optimization to effectively address parameter variations and maintain the efficiency required for real-time decision-making processes.

5.1 Configuration of test instances

Our computational experiments are conducted using test instances constructed from an analysis of drilling data from a coal mine, specifically focusing on the overburden removal process. Table 1 summarizes key characteristics of our case study.

Table 1: Summary of Case Study Information

Characteristic	Value
Data timespan	9 months
Mine type	Coal mine
Maximum active machines per day	3
Average travel time between consecutive tasks (same column)	1 minute

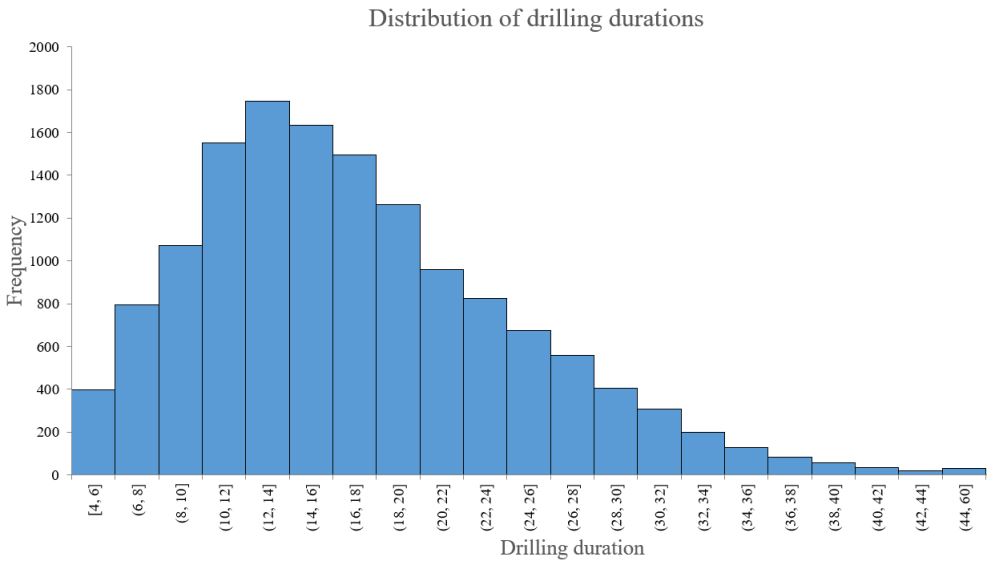
To calculate the travel time between any pair of tasks, we utilize a formula that considers both vertical and horizontal movements within the task layout. This formula helps us determine how much time it takes to move from one task to another, accounting for the relative positions of tasks in the drilling pattern.

To simulate the drilling times, we generate random numbers that follow the observed distribution over the nine-month period. This simulation process involves calculating the cumulative density

function (CDF) of the recorded drilling times. We then generate a set of uniform random numbers between 0 and 1, with the number of generated values matching the number of target-machine pairs. By evaluating the CDF at each generated number, we select corresponding values from the available data, ensuring that the probability aligns with the generated value. The distribution of the drilling times is visually illustrated in Figure 9. In this graphical representation, the x-axis represents the range of drilling times observed, divided into discrete intervals, while the y-axis represents the frequency or count of observations falling into each interval.

It is important to note that during the data analysis, we did not exclude drilling times that are considered outliers. These outliers represent machine failures that occurred during the nine months, and we aim to simulate drilling operations as accurately as possible.

Figure 9: The distribution of the observed drilling times (in minutes)



In the context of our industrial partner's operations, they adopt a batching approach for drilling activities. Specifically, they complete batches consisting of up to 300 targets within a timeframe of 3 to 4 days, which translates to approximately 100 tasks per day, as indicated by the data. Additionally, our partner follows a safety protocol that enforces a distance of two columns between operating machines.

It is important to note that the test instances we generate intentionally push the boundaries of the model by incorporating larger numbers of targets compared to the actual parameters provided by our industrial partner. Furthermore, the time is discretized into 1-minute intervals, with each time period representing 1 minute. These choices aim to showcase the model's limitations and enable a comprehensive evaluation of its performance in challenging scenarios.

5.2 Computational results

Table 2 presents the computational results obtained from our CP model for instances with 3 machines and a planning horizon of 24 hours. The first two columns, r and c, denote the number of rows and columns, respectively, within the drilling pattern. It is important to note that the number of tasks, n, is the product of r and c. For each combination of r and c, we generate 20 random instances based on the distribution of task processing times. The remaining columns display averaged performance metrics derived from the 20 instances with the same r and c values. The "Objective Function" column represents the average number of tasks scheduled by the model within the given timeframe. The "Standard Deviation" column indicates the standard deviation of the objective function values.

Lastly, the "O-Gap (%)" represents the optimality gap, which is the relative difference between the best feasible solution found and the best known upper bound on the optimal solution.

Table 2: Instances with m = 3 machines and a planning horizon of H = 24 hours

r	c	Objective Function	Standard Deviation	O-Gap (%)
10	30	284.25	5.64	5.58
10	25	250	0	0
10	20	200	0	0
10	15	150	0	0
10	10	100	0	0

Table 2 provides valuable insights into the behavior of the proposed CP model and its ability to solve instances of varying complexity. It demonstrates that for configurations with fewer than 250 tasks, the model achieves optimal solutions with no deviations, as indicated by zero standard deviation and gap. However, for instances with 300 tasks, we observe slight deviations and variability. On average, the model schedules up to 284 tasks within a 24-hour planning period while staying within the solver's 2-minute runtime limit.

The ability to consistently and quickly find near-optimal solutions makes the proposed model suitable for real-time decision-making. This is particularly valuable in practical scenarios where the processing time for each task is not accurately known at the beginning of drilling operations. In such cases, the model can be used to create a drilling schedule based on estimated processing times computed using historical data. As drilling progresses, the schedule can be adjusted by re-solving the model based on updated estimates of processing times over shorter planning horizons.

Table 3 is designed to showcase the limits of the model, providing valuable insights into its performance under different configurations. The table displays the computational results of the model for instances with 2 machines and planning periods ranging from 1 to 3 days, as well as instances with 3 machines and planning periods of 1 and 2 days. For each configuration, the results from the most challenging instance are presented, allowing us to evaluate the model's ability to handle more complex scenarios. The first three columns, "Tasks", "m", and "H", indicate the number of tasks, the number of available machines, and the planning horizon, respectively. The next three columns, "Objective Function", "Standard Deviation", and "O-Gap (%)", present the same averaged performance metrics as shown in Table 2. The last column, "H-Gap (%)", presents the average gap between the solution found by the solver after 2 minutes of run time and the initial solution provided by the heuristic algorithm presented in Section 4.2.

Table 3: Computational results

Tasks	m	H	Objective Function	Standard Deviation	O-Gap (%)	H-Gap (%)
200	2	24	196.6	2.09	1.74	21.99
400	2	48	385.1	8.04	3.91	25.34
600	2	72	555.2	18.50	8.18	22.08
300	3	24	284.25	5.64	5.58	23.41
600	3	48	491.85	40.47	22.72	15.06

The standard deviation, reflecting the variability in the objective function values across the 20 generated instances for each configuration, generally increased with the number of tasks. This implies greater variability in the model's performance as the complexity of the instances heightened. However, it is important to note that even for the challenging instances with 600 tasks, the standard deviation values, such as 18.50 and 40.47, are still relatively moderate, representing 3.3% and 8.2% of their respective averages. This suggests that the model maintains a certain level of consistency and reliability, despite the increased difficulty in scheduling a larger number of tasks.

The optimality gap percentage, which measures the average discrepancy between the best solution obtained and the upper bound on the optimal solution, provides further insights. Instances with higher numbers of tasks and longer planning periods exhibited larger gap percentages. While the larger instances demonstrated larger gaps, it is important to consider that these instances go beyond the scale of practical scenarios. Therefore, the observed gaps should be interpreted with caution, as they represent extreme scenarios that are not typically encountered in real-world drilling operations.

The H-Gap column showcases the percentage improvement of the CP model's solution over the initial heuristic solution. On average, the CP model schedules approximately 20% more tasks than the heuristic algorithm. This substantial difference underscores the significant advantages of the CP model. For instance, in the case of 300 tasks with 3 machines over 24 hours, the CP model schedules 23.41% more tasks than the heuristic. This improvement translates to a significant increase in operational efficiency, demonstrating the value of the CP approach in practical drilling operations.

In summary, the computational results presented in Table 3 demonstrate the model's effectiveness in scheduling tasks within practical ranges. The model performs exceptionally well for instances representing the task volumes commonly encountered in real-world scenarios. While some challenges arise when dealing with instances containing a significantly larger number of tasks, these instances go beyond the scope of practical implementations. Therefore, the observed limitations should be viewed in the context of the model's applicability to real-world drilling operations. The findings provide valuable insights for further research and development efforts to enhance the model's performance and optimize drilling activities under a variety of practical conditions.

6 Conclusion

In the field of mining, drilling plays a crucial role in the overall operation. As the industry moves towards electrification and automation to meet sustainability goals, optimizing the coordination of electrical drill rigs becomes increasingly important. While combinatorial optimization has been extensively applied to various aspects of mining, the optimization community has largely overlooked the important task of detailed drill scheduling. Existing studies have focused on improving individual drills or integrating drilling into broader production schedules without considering the day-to-day coordination of the entire drilling process. This research addresses this gap by introducing a scheduling problem that arises in the context of coordinating multiple electrical drill rigs which operate in close proximity within open-pit mines.

The unique characteristics of the problem include spatial and temporal constraints, where task availability depends on the current location of the drill rigs and collision avoidance constraints must be satisfied. To tackle this problem, we provide both mixed-integer programming and constraint programming formulations.

Our computational experiments, as presented in Tables 2 and 3, demonstrate the efficiency of the proposed CP model in solving large-scale instances encountered in industrial settings. For example, the model successfully schedules up to 284 out of 300 tasks within a 24-hour period using 3 machines, with an optimality gap of only 5.58%. The model's capacity to deliver near-optimal solutions within a 2-minute timeframe presents a distinct advantage for real-time decision making. This attribute proves particularly valuable given the inherent uncertainty associated with task processing times, which gradually become more accurately known as drilling operations progress. Additionally, the schedules generated by the model offer detailed, minute-by-minute information about the operation of electrical drill rigs, as evidenced by our ability to discretize time into 1-minute intervals, enabling precise planning and coordination, further enhancing operational efficiency.

Furthermore, considering the uncertain nature of processing times in practice, future studies should explore the development of robust scheduling approaches that can handle stochastic scenarios. By accounting for the unknown nature of processing times, the scheduling solutions can be more reliable and

adaptable to real-world uncertainties. Additionally, integrating this short-term scheduling approach with medium and long-term planning models could provide a more comprehensive solution for mining operations.

Overall, this research contributes to the ongoing efforts to make mining operations more efficient, sustainable, and automated. It sheds light on the importance of addressing the drilling scheduling problem in mining operations and provides valuable insights, models, and directions for future research to optimize and improve the efficiency of drilling processes in the context of modern, electrified mining operations.

7 Declaration of interest

The authors report there are no competing interests to declare.

References

Afeni, T. B. (2009). Optimization of drilling and blasting operations in an open pit mine the SOMAIR experience. *Mining Science and Technology (China)*, *19*(6), 736–739. doi: 10.1016/S1674-5264(09)60134-4

Alarie, S., & Gamache, M. (2002). Overview of Solution Strategies Used in Truck Dispatching Systems for Open Pit Mines. *International Journal of Surface Mining, Reclamation and Environment*, *16*(1), 59–76. doi: 10.1076/ijsm.16.1.59.3408

Azarafza, M., Feizi Derakhshi, M. R., & Jeddi, A. (2017, 09). Blasting pattern optimization in open-pit mines by using the genetic algorithm. *Geotechnical Geology*, *13*, 75–81.

Bao, H., Knights, P., Kizil, M., & Nehring, M. (2023, January). Electrification Alternatives for Open Pit Mine Haulage. *Mining*, *3*(1), 1–25. Retrieved 2024-09-19, from https://www.mdpi.com/2673-6489/3/1/1 doi: 10.3390/mining3010001

Barnewold, L., & Lottermoser, B. G. (2020, November). Identification of digital technologies and digitalisation trends in the mining industry. *International Journal of Mining Science and Technology*, *30*(6), 747–757. Retrieved 2024-09-19, from https://linkinghub.elsevier.com/retrieve/pii/S2095268620300744 doi: 10.1016/j.ijmst.2020.07.003

Chowdu, A., Nesbitt, P., Brickey, A., & Newman, A. M. (2022, March). Operations Research in Underground Mine Planning: A Review. *INFORMS Journal on Applied Analytics*, *52*(2), 109–132. Retrieved 2024-09-19, from https://pubsonline.informs.org/doi/10.1287/inte.2021.1087 doi: 10.1287/inte.2021.1087

Dutaut, R., & Marcotte, D. (2020, December). A New Semi-greedy Approach to Enhance Drillhole Planning. *Natural Resources Research*, *29*(6), 3599–3612. Retrieved 2024-08-04, from https://link.springer.com/10.1007/s11053-020-09674-8 doi: 10.1007/s11053-020-09674-8

Fathollahzadeh, K., Asad, M. W. A., Mardaneh, E., & Cigla, M. (2021). Review of Solution Methodologies for Open Pit Mine Production Scheduling Problem. *International Journal of Mining, Reclamation and Environment*, *35*(8), 564–599. doi: 10.1080/17480930.2021.1888395

Hosseini, S., Monjezi, M., & Bakhtavar, E. (2022). Minimization of blast-induced dust emission using gene-expression programming and grasshopper optimization algorithm: a smart mining solution based on blasting plan optimization. *Clean Technologies and Environmental Policy*, *24*, 2313 - 2328. Retrieved from https://api.semanticscholar.org/CorpusID:248519985

Hosseini, S., Mousavi, A., Monjezi, M., & Khandelwal, M. (2022). Mine-to-crusher policy: Planning of mine blasting patterns for environmentally friendly and optimum fragmentation using monte carlo simulation-based multi-objective grey wolf optimization approach. *Resources Policy*, *79*, 103087. doi: https://doi.org/10.1016/j.resourpol.2022.103087

Hossein-Morshedy, A., Khorram, F., & Emery, X. (2023, September). A Multi-Objective Approach for Optimizing the Layout of Additional Boreholes in Mineral Exploration. *Minerals*, *13*(10), 1252. Retrieved 2024-08-04, from https://www.mdpi.com/2075-163X/13/10/1252 doi: 10.3390/min13101252

Inanloo Arabi Shad, H., Sereshki, F., Ataei, M., & Karamoozian, M. (2018). Prediction of rotary drilling penetration rate in iron ore oxides using rock engineering system. *International Journal of Mining Science and Technology*, *28*(3), 407–413. doi: 10.1016/j.ijmst.2018.04.004

Jafrasteh, B., & Suárez, A. (2021, February). Objective functions from Bayesian optimization to locate additional drillholes. *Computers & Geosciences*, *147*, 104674. Retrieved 2024-08-04, from https://linkinghub.elsevier.com/retrieve/pii/S0098300420306464 doi: 10.1016/j.cageo.2020.104674

Jaswani, P., & Jeffs, J. (2024). *Electric vehicles in mining 2024-2044: Technologies, players, and forecasts* (Market Research Report). IDTechEx. Retrieved from https://www.idtechex.com/en/research-report/electric-vehicles-in-mining-2024-2044-technologies-players-and-forecasts/994 (Accessed on September 26, 2024)

Kozan, E., & Liu, S. (2016). A new open-pit multi-stage mine production timetabling model for drilling, blasting and excavating operations. *Transactions of the Institutions of Mining and Metallurgy, Section A: Mining Technology*, *125*(1), 47–53. Retrieved from http://dx.doi.org/10.1179/1743286315Y.0000000031

Kozan, E., & Liu, S. Q. (2011). Operations research for mining: a classification and literature review. *ASOR Bulletin*, *30*(1), 2–23.

Kozan, E., & Liu, S. Q. (2017). An operational-level multi-stage mine production timetabling model for optimally synchronising drilling, blasting and excavating operations. *International Journal of Mining, Reclamation and Environment*, *31*(7), 457–474. Retrieved from http://dx.doi.org/10.1080/17480930.2016.1160818

Laborie, P., Rogerie, J., Shaw, P., & Vilím, P. (2018). IBM ILOG CP optimizer for scheduling: 20+ years of scheduling with constraints at IBM/ILOG. *Constraints*, *23*(2), 210–250. doi: 10.1007/s10601-018-9281-x

Leung, R., Hill, A. J., & Melkumyan, A. (2023). Automation and Artificial Intelligence Technology in Surface Mining: A Brief Introduction to Open-Pit Operations in the Pilbara. *IEEE Robotics & Automation Magazine*, 2–21. Retrieved 2024-09-19, from https://ieeexplore.ieee.org/document/10328439/ doi: 10.1109/MRA.2023.3328457

Maleki, M., Baeza, D., Soltani-Mohammadi, S., Madani, N., Díaz, E., & Anguita, F. (2024). Optimising the placement of additional drill holes to enhanced mineral resource classification: a case study on a porphyry copper deposit. *International Journal of Mining, Reclamation and Environment*, 1–18. Retrieved from https://www.tandfonline.com/doi/full/10.1080/17480930.2024.2364131 doi: 10.1080/17480930.2024.2364131

Mansouri, M., Andreasson, H., & Pecora, F. (2016). Hybrid reasoning for multi-robot drill planning in open-pit mines. *Acta Polytechnica*, *56*(1), 47. doi: 10.14311/APP.2016.56.0047

Mansouri, M., Lagriffoul, F., & Pecora, F. (2017). Multi vehicle routing with nonholonomic constraints and dense dynamic obstacles. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 3522–3529). Vancouver, BC: IEEE. doi: 10.1109/IROS.2017.8206195

Moradi Afrapoli, A., & Askari-Nasab, H. (2019). Mining fleet management systems: a review of models and algorithms. *International Journal of Mining, Reclamation and Environment*, *33*(1), 42–60. doi: 10.1080/17480930.2017.1336607

Mwangi Akisa, D., Huang, G., Jianhua, Z., Kasomo, R., & Matidza, M. (2021). Ultimate pit limit optimization methods in open pit mines: A review. *Journal of Mining Science*, *56*, 588–602. doi: 10.1134/S1062739120046885

Nageshwaraniyer, S., Kim, K., & Son, Y.-J. (2015, 11). Optimal blast design using a discrete-event simulation model in a hard-rock mine. *Mining Engineering*, *67*, 47–53.

Newman, A. M., Rubio, E., Caro, R., Weintraub, A., & Eurek, K. (2010). A Review of Operations Research in Mine Planning. *Interfaces*, *40*(3), 222–245. doi: 10.1287/inte.1090.0492

Onifade, M., Said, K. O., & Shivute, A. P. (2023, July). Safe mining operations through technological advancement. *Process Safety and Environmental Protection*, *175*, 251–258. Retrieved 2024-09-19, from https://linkinghub.elsevier.com/retrieve/pii/S0957582023004202 doi: 10.1016/j.psep.2023.05.052

Onifade, M., Zvarivadza, T., Adebisi, J. A., Said, K. O., Dayo-Olupona, O., Lawal, A. I., & Khandelwal, M. (2024, June). Advancing toward sustainability: The emergence of green mining technologies and practices. *Green and Smart Mining Engineering*, *1*(2), 157–174. Retrieved 2024-09-19, from https://linkinghub.elsevier.com/retrieve/pii/S2950555024000338 doi: 10.1016/j.gsme.2024.05.005

Osanloo, M., Gholamnejad, J., & Karimi, B. (2008). Long-term open pit mine production planning: a review of models and algorithms. *International Journal of Mining, Reclamation and Environment*, *22*(1), 3–35. doi: 10.1080/17480930601118947

Rais, K., Kara, M., Riheb, H. H., Gadri, L., & Khochmen, L. (2017). Original approach for the drilling process op-timization in open cast mines; case study of Kef Essenoun open pit mine northeast of Algeria. *Mining Science*. doi: 10.5277/MSC172409

Servet, D., Nazmi, S., Ibrahim, U., Tamer, E., Deniz, A., & Rasit, A. (2014). Variation of vertical and horizontal drilling rates depending on some rock properties in the marble quarries. *International Journal of Mining*

Science and Technology, *24*(2), 269–273. doi: 10.1016/j.ijmst.2014.01.020

Shangxin, F., Yujie, W., Guolai, Z., Yufei, Z., Shanyong, W., Ruilang, C., & Enshang, X. (2020). Estimation of optimal drilling efficiency and rock strength by using controllable drilling parameters in rotary non-percussive drilling. *Journal of Petroleum Science and Engineering*, *193*, 107376. doi: 10.1016/j.petrol.2020.107376

Shen, Q., Wang, Y., Cao, R., & Liu, Y. (2022). Efficiency evaluation of a percussive drill rig using rate-energy ratio based on rock drilling tests. *Journal of Petroleum Science and Engineering*, *217*, 110873. Retrieved from https://www.sciencedirect.com/science/article/pii/S092041052200729X doi: https://doi.org/10.1016/j.petrol.2022.110873

Ugurlu, O. F., & Kumral, M. (2020a). Cost optimization of drilling operations in open-pit mines through parameter tuning. *Quality Technology & Quantitative Management*, *17*(2), 173–185. doi: 10.1080/16843703.2018.1564485

Ugurlu, O. F., & Kumral, M. (2020b). Management of Drilling Operations in Surface Mines Using Reliability Analysis and Discrete Event Simulation. *Journal of Failure Analysis and Prevention*, *20*(4), 1143–1154. doi: 10.1007/s11668-020-00921-x

Uğurlu, O. F. (2021). Drill bit monitoring and replacement optimization in open-pit mines. *Bilimsel Madencilik Dergisi*. doi: 10.30797/madencilik.847142

van den Akker, J. M., Hoogeveen, J. A., & van Kempen, J. W. (2006). Parallel machine scheduling through column generation: minimax objective functions. In D. Hutchison et al. (Eds.), *Algorithms – ESA 2006* (Vol. 4168, pp. 648–659). Berlin, Heidelberg: Springer Berlin Heidelberg. doi: 10.1007/11841036_58

Zagré, G. E., Marcotte, D., Gamache, M., & Guibault, F. (2019, July). New Tabu Algorithm for Positioning Mining Drillholes with Blocks Uncertainty. *Natural Resources Research*, *28*(3), 609–629. Retrieved 2024-08-04, from http://link.springer.com/10.1007/s11053-018-9412-5 doi: 10.1007/s11053-018-9412-5