Contemporary Mathematics
https://ojs.wiserpub.com/index.php/CM/

Research Article

Enhanced Short-Term Scheduling of Underground Mining Activities Using Tabu Search: A Comparative Analysis

Luis Álvarez[1], Heber Hernández[1], Elisabete Alberdi[2], Aitor Goti[3*]

[1] Civil Mining Engineering, Faculty of Engineering, Santo Tomás University, Liberation Army 146 (Ingeniería Civil en Minas, Facultad de Ingeniería, Universidad Santo Tomás, Ejército Libertador 146), Santiago, 8370003, Chile
[2] Department of Applied Mathematics, University of the Basque Country (UPV/EHU), Rafael Moreno 'Pitxitxi', 2, Bilbao, 48013, Spain
[3] Department of Mechanics, Design and Industrial Management, Faculty of Engineering, University of Deusto, Avda. de las Universidades, Bilbao, 48007, Spain
E-mail: aitor.goti@deusto.es

Received: 22 August 2025; **Revised:** 16 September 2025; **Accepted:** 22 September 2025

Abstract: The planning of preparation and development activities in underground mining is essential to ensure efficiency and operational continuity. However, short-term scheduling of these tasks has received limited attention in literature. This study proposes a Cooperative Multi-start Tabu Search with Path-Relinking (CMTS-PR), which coordinates multiple tabu trajectories and intensifies them through path relinking to optimize the short-term scheduling of multiple underground heading works within a one-shift horizon. The problem is modeled as a flexible job-shop scheduling problem with unrelated parallel equipment and sequence dependent setup times. CMTS-PR is evaluated against a memetic algorithm, a Non-dominated Sorting Genetic Algorithm II, a single-trajectory Tabu Search (TS), a Constraint Programming (CP) model, and manual scheduling by an expert planner, across two panel caving case studies in Chile. The results show that CP yields mathematically optimal solutions but becomes computationally demanding, while manual scheduling ensures feasibility but underutilizes resources. In contrast, CMTS-PR produces operationally viable schedules. In case study 1, CMTS-PR matched CP on equivalent fronts within 60 seconds, even under 10-60 minute transfer time variability. In case study 2, CMTS-PR increased equivalent fronts by 120% compared to manual planning and by 2.94% relative to CP, with lower runtime. Overall, CMTS-PR proves to be effective and computationally efficient, representing one of the first applications of a cooperative TS and path-relinking scheme to underground short-term scheduling, and providing a practical tool for daily mine operations.

Keywords: underground short-term scheduling, flexible job shop scheduling, tabu search, genetic algorithms

MSC: 90B35, 90C90, 90C11

Abbreviation

CMTS-PR	Cooperative Multi-start Tabu Search with Path-Relinking
CP	Constraint Programming
CSV	Comma-Separated Value

NCCoC	National Copper Corporation of Chile
JSSP	Job Shop Scheduling Problem
FJSSP	Flexible Job Shop Scheduling Problem
GA	Genetic Algorithm
MILP	Mixed Integer Linear Programming
MUHW	Multiple Underground Headings Work
NSGA-II	Non-dominated Sorting Genetic Algorithm II
SDS	Sequence Dependent Setups
TS	Tabu Search
UHW	Underground Headings Work

1. Introduction

Underground mining is requiring the construction of a complex network of excavations developed in the rock mass, strategically designed to enable the efficient and safe extraction of minerals [1–4]. This process is based on geotechnical engineering principles, which ensure structural stability and minimize risks during operations [5]. Mine planning in this context is primarily oriented toward optimizing production within long, medium, and short-term geotechnical and operational constraints [6–8]. However, mine development and preparation, which include the construction of horizontal and vertical drifts and other civil works necessary to enable extraction zones according to the production plan, are often planned in a secondary stage and lack adequate optimization [9]. Recent research has made significant progress in integrating production and preparation optimization over medium and long-term planning horizons [10–13]. Nevertheless, the optimization of short-term planning remains largely unexplored. Short-term planning focuses on operational decisions within a reduced time horizon, ranging from one-day to two years [14, 15]. Short-term planning includes tasks such as production scheduling, equipment allocation and maintenance, and defining extraction and destination decisions, ensuring compliance with operational constraints, production goals and construction deadlines [16, 17]. This planning seeks to coordinate a wide variety of resources, from workers to equipment at Multiple Underground Heading Works (MUHW) and must address diverse limitations inherent to each of these resources [18]. According to Topal [19], this task becomes particularly complex due to the multiple possible combinations for assigning resources to activities, resulting in many options to evaluate. Moreover, short-term planning constraints are highly interdependent, further complicating the process. For instance, equipment operation times, worker shifts, and resource availability depend on the activity sequence and can mutually influence each other. Furthermore, mining activities are closely interconnected, meaning that a change in one activity can directly impact others, complicating process management and optimization. For example, constructing a horizontal drift involves a sequence of activities that must be carried out in an orderly manner. This set of tasks is known as a development cycle [20], and in mines exploited using the panel caving method, which will serve as the example in this application. This development cycle follows the sequence shown in Figure 1.

According to Andrade and Rampazzo [21] and Aalian et al. [22], underground mining shift planning is traditionally done manually with spreadsheet support, which does not guarantee optimal process outcomes [23–25]. It often varies depending on the planner's experience, introducing subjectivity. Manual planning is time-consuming, limiting the number of options considered [26, 27]. A shift schedule must include at least the start and end times for each work heading activity, as well as the machine assigned to each stage of the cycle [28]. This schedule determines the number of blasts achievable during a shift. It is important to note that the distance between MUHW directly affects performance. However, this time is typically integrated as a distinct average for each activity within the cycle.

Mixed-Integer Linear Programming (MILP) models have been widely used for scheduling activities in mining, particularly during production optimization. For example, Schulze et al. [29] scheduled the mobile production fleet in a potash mine, using a commercial solver for small instances and developing heuristics to improve scalability. Similarly, O'Sullivan and Newman [8] worked on the Lisheen mine (Ireland), maximizing mineral production and incorporating a decomposition heuristic to make the problem more computationally manageable. Martinez and Newman [30] applied a decomposition strategy based on ore grades in the Kiruna mine (Sweden), optimizing long-term resource allocation.

Nehring et al. [7] optimized equipment allocation in a mine using sublevel stoping, minimizing deviations from the target production. These studies confirm MILP as a reference method for mine scheduling. Nevertheless, they reveal a recurring dependence on complementary heuristics to manage computational complexity in large-scale applications.

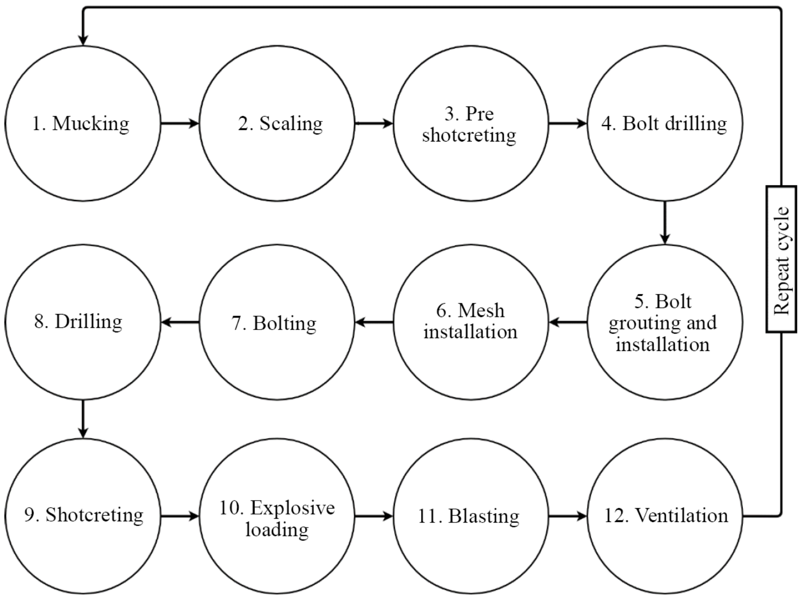

Figure 1. Mining development cycle operations for a horizontal drift in panel caving method

More recently, Constraint Programming (CP) approaches have shown strong potential for short-term underground scheduling, capturing operational constraints and travel times with high fidelity [22, 25, 28]. These works highlight CP's capacity to outperform manual planning and to generate robust schedules even under uncertainty. In parallel, Genetic Algorithms (GA) have also been applied across several mining contexts. Andrade and Rampazzo [20] optimized the Job-Shop Scheduling Problem (JSSP) at the Cuiabá gold mine using a GA; Wang et al. [31] addressed trackless equipment scheduling; and Miao and Zhao [32] proposed an improved GA for ramp-congestion scheduling. Collectively, these studies underscore GA's adaptability to diverse mine operations. Building on this line of work, the Non-dominated Sorting Genetic Algorithm II (NSGA-II) has become a strong baseline for (flexible) job-shop variants, efficiently approximating Pareto fronts over makespan, energy/cost, and emissions, often with memetic or light local search intensification [33]. Complementing these evolutionary approaches, Memetic Algorithms (MA), combining genetic search with embedded local improvement, have also yielded strong results for the Flexible Job-Shop Scheduling Problem (FJSSP), including a multi objective scheme that balances makespan and machine workload. By contrast, Tabu Search (TS) has been scarcely explored in underground short-term scheduling, despite strong evidence in related domains. In open pit contexts, TS has delivered high quality solutions under geological uncertainty for example, through diversified TS with advanced diversification strategies [34] and comparative studies confirm its industrial scale competitiveness for short-term block sequencing [35]. Beyond mining, a substantial scheduling literature validates TS on complex combinatorial problems. Early contributions include Brandimarte [36], one of the first TS formulations for the FJSSP, and Hurink et al. [37], who extended TS to multi-purpose machines. More recently, TS has been studied in FJSSP variants with sequence dependent setups and unrelated parallel machines, targeting tardiness and makespan. Building on this trajectory, the literature also explores hybrid and cooperative TS designs, most notably TS with Path-Relinking (TS/PR), which relinks elite solutions to intensify search, as well as multi-start memetic enhancements that balance diversification and intensification across. These advances motivate a cooperative design tailored to underground FJSSP with sequence dependent setups. Together, these works illustrate TS's versatility and robustness; however, its direct application to underground development cycles over short horizons remains largely unexplored, precisely the gap we address. In this study, we therefore consider four complementary baselines: (i) CP, a declarative paradigm that encodes variables, domains, and constraints and can prove optimality on small/medium instances; (ii) MA, a population based metaheuristic widely used in mining for its exploration and exploitation balance [38]; (iii) NSGA-II, a well-established evolutionary baseline for flexible job-shop variants that efficiently approximates Pareto fronts; and (iv) a single trajectory TS reference. This set provides a rigorous yardstick against which to assess the contribution of a cooperative TS approach.

This article presents a Cooperative Multi-start Tabu Search with Path-Relinking (CMTS-PR) for short-term scheduling of horizontal drift activities in underground mines. Unlike conventional approaches that treat equipment transfer times between MUHW as fixed constants, we model sequence-dependent setups to capture spatial logistics. The problem is cast as an FJSSP with unrelated parallel equipment, a configuration seldom addressed in underground settings. To tackle NP-hardness, CMTS-PR launches multiple tabu trajectories in parallel and coordinates them via an elite-set path-relinking mechanism, balancing diversification across trajectories and intensification around high-quality regions. We benchmark CMTS-PR against CP, MA, NSGA-II, and single-trajectory TS, using the same inputs (cycle activity times, equipment availability, effective shift duration, and variable transfer times) and reporting comparable outputs (complete schedules with Gantt visualization, makespan, and equivalent fronts).

1.1 *Structure of the paper*

The rest of the paper is structured as follows. Section 2 presents the methodology, including a detailed definition of the input variables, the formulation of the problem as an FJSSP model, the objective function, and the associated constraints. Additionally, we formally define the solver suite, incorporating TS, MA, NSGA-II, and our novel CMTS-PR, alongside CP and MILP baselines. Section 3 presents two case studies of underground mining operations utilizing the panel caving method to validate the proposed methodology. Section 4 presents the results of both case studies, evaluating the performance of each method. Section 5 discusses key insights, including the prioritization of Underground Heading Works (UHW), the incorporation of variable transfer times, modeling equipment failures, and addressing operational uncertainty. Finally, Section 6 concludes the paper by summarizing the contributions and outlining directions for future research.

2. Methodology

The problem to be solved involves determining the optimal scheduling of activities for the horizontal drift cycle in underground mining within a one-day time horizon. This scheduling must consider the travel time of machinery between MUHW, as current short-term planning assumes this time to be a fixed value added to the duration of each activity, without considering the actual distances between each UHW. Wang et al. [31] propose incorporating setup time as part of the model to integrate equipment travel times into the FJSSP.

2.1 *Inputs variables definition*

The input data for this problem includes:

• Durations of activities for each stage of the mining cycle (e.g., drilling, blasting, mucking, bolting).

• The availability of equipment for each activity.

• The effective shift duration.

• The travel times between each UHW, modeled as sequence-dependent setup times.

This constitutes the basic information that can be incorporated into this methodology (Figure 2). The discussion section of the article provides details on potential new sources of information and how they can be integrated into the approach.

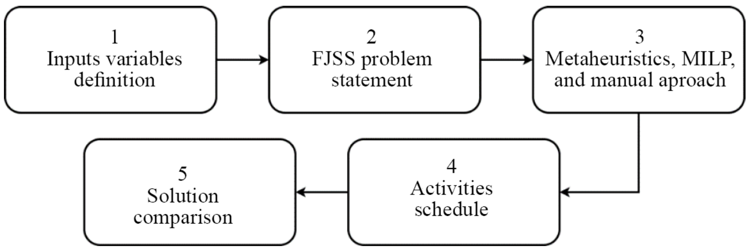
1
Inputs variables definition
2
FJSS problem statement
3
Metaheuristics, MILP, and manual aproach
4
Activities schedule
5
Solution comparison

Figure 2. Methodology outline

2.2 *FJSSP statement*

The short-term scheduling of UHW is modeled as a FJSSP. The problem consists of a set of jobs J (the UHW), where each job $j \in J$, requires a sequence of operations (e.g., drilling, bolting). A set of machines M is available, and the "flexibility" arises because each operation (j, t) can be processed by a specific subset of eligible machines, denoted as $M_{j,t} \subseteq M$. The objective is to assign a machine to each operation and sequence them to optimize a performance metric, typically minimizing the makespan ($C_{\max}$).

2.3 *Equivalent underground headings*

To assess the effectiveness of the proposed methodology, two performance metrics are employed:

• Makespan ($C_{\max}$): the total completion time of the schedule, representing the overall duration required to finalize all tasks within the planning horizon.

• Equivalent Underground Headings Index (F_{eq}):

$$F_{eq} = \sum_{k \in \Lambda} \alpha_p, \quad \alpha_{kp} = \frac{t_p}{\sum_l t_l} \tag{1}$$

where t_p is the duration of activity p, and Λ is the set of completed activities. This indicator represents the effective number (possibly fractional) of underground headings completed under a given schedule. These metrics allow for a comprehensive assessment of both time efficiency and operational progress, enabling a fair comparison between exact optimization, metaheuristics, and manual scheduling practices.

2.4 *Constraint programming*

This model formulates the FJSSP with unrelated parallel machines and sequence dependent setup times using CP, specifically the CP-SAT (Boolean Satisfiability) solver. This approach encodes each task as a set of alternative optional intervals, one per eligible machine. Sequence-dependent setups are modeled via pairwise precedence decisions between tasks scheduled on the same machine.

2.4.1 *Sets and parameters*

• J: Set of jobs $\{1, 2, \ldots, n_j\}$.
• T_j: Set of tasks associated with job j.
• $M_{j,t}$: Set of available machines for task t of job j.
• $p_{j,t}$: Processing time of task (j, t) independent of machine m.
• $s_{(j,t),(j',t')}$: Setup time between tasks (j, t) and (j', t') if processed sequentially on the same machine.
• H: Scheduling horizon; a sufficiently large value to upper bound all time-related variables.

2.4.2 *Decision variables*

- $x_{j,\,t,\,m} \in \{0,\,1\}$: Binary variable equal to 1 if task $(j,\,t)$ is assigned to machine m, and 0 otherwise.
- $s_{j,\,t} \in \{0,\,H\}$: Start time of task $(j,\,t)$.
- $e_{j,\,t} \in \{0,\,H\}$: End time of task $(j,\,t)$.
- $o_{(j,\,t),\,(j',\,t'),\,m} \in \{0,\,1\}$: Binary variable equal to 1 if task $(j,\,t)$ precedes task $(j',\,t')$ on machine m.
- $makespan \in \{0,\,H\}$: Total completion time of the last job in the schedule.

2.4.3 *Constraints*

Each task must be processed by exactly one machine from its eligible set:

$$\sum_{m\in M} x_{j,\,t,\,m} = 1, \quad \forall j \in J, \quad \forall t \in T_j \tag{2}$$

The completion time of each task is defined as:

$$e_{j,\,t} = s_{j,\,t} + p_{j,\,t}, \quad \forall j \in J, \quad \forall t \in T_j. \tag{3}$$

Each task in a job must start only after the completion of its predecessor:

$$s_{j,\,t+1} \geq e_{j,\,t}, \quad \forall j \in J, \quad \forall t \in T_j \setminus \left\{t_{\max j}\right\} \tag{4}$$

where $t_{\max j}$ is the last task of job j. If two distinct tasks are assigned to the same machine, they must be sequenced with the appropriate setup time:

$$s_{j',\,t'} \geq e_{j,\,t} + s_{(j,\,t),\,(j',\,t')} \cdot o_{(j,\,t),\,(j',\,t'),\,m}, \quad \forall m, \quad \forall (j,\,t) \neq \left(j',\,t'\right) \tag{5}$$

Symmetrically, the alternative condition is enforced:

$$s_{(j,\,t)} \geq e_{(j',\,t')} + s_{(j',\,t')(j,\,t)} \cdot \left(1 - o_{(j,\,t),\,(j',\,t'),\,m}\right), \quad \forall m, \quad \forall (j,\,t) \neq \left(j',\,t'\right) \tag{6}$$

These constraints ensure that:

$$\begin{cases} O_{(j,\,t),\,(j',\,t')},\, m = 1 \text{ means task } (j,\,t) \text{ precedes task } \left(j',\,t'\right) \text{ on machine } m \\ \\ O_{(j,\,t),\,(j',\,t')},\, m = 0 \text{ means the opposite order.} \end{cases} \tag{7}$$

We define in Equation (8) the makespan $C_{\max}$ as the maximum job completion time. Let $t_{last(j)}$ denote the last operation of job j. We enforce the link with operation completion times and set the objective to minimize the makespan.

$$C_{\max} \geq e_{j,\, t_{\text{last}\,(j)}} \forall j J, \tag{8}$$

2.5 *MILP*

We formulate the FJSSP with alternative machines and sequence dependent setup times as a mixed-integer linear program. Start times are continuous; each operation is assigned to exactly one eligible machine; intra-job precedences are enforced on start times; and machine level no overlap is modeled with Big-M disjunctive constraints and binary ordering variables for every task pair on the same machine, adding the sequence-dependent setup between consecutive tasks. The objective minimizes the makespan $C_{\max}$. We solve the model with the COIN Branch and Cut solver (CBC) MILP solver vía Python Linear Programming (PuLP) library.

2.5.1 *Sets and parameters*

- J: Set of jobs $\{1, 2, \ldots, n_j\}$.
- T_j: Set of tasks of job j.
- $M_{j,\,t}$: Set of available machines for task (j, t).
- $p_{j,\,t}$: Processing time of (j, t) independent of machine m.
- $s_{(j,\,t),\,(j',\,t')}$: Setup time if (j, t) precedes (j', t') on the same machine.
- H: Scheduling horizon.
- $P_m = \{(j, t) : m \in M_{j,\,t}\}$: tasks that can be processed on machine m.

2.5.2 *Decision variables*

- $x_{j,\,t,\,m} \in \{0, 1\}$: equals to 1 if task (j, t) is assigned to machine m, and 0 otherwise.
- $s_{j,\,t} \in \{0, H\}$: Start time of task (j, t).
- $e_{j,\,t} \in \{0, H\}$: End time of task (j, t).
- $y_{i,\,k,\,m} \in \{0, 1\}$: equals to 1 if task i precedes task k on machine m $(i \neq k,\, i,\, k \in P_m)$.
- $C_{\max} \in \{0, H\}$: makespan.

2.5.3 *Constraints*

Each task must be processed by exactly one machine from its eligible set:
The completion time of each task is defined as:

$$\sum_{m \in M} x_{j,\,t,\,m} = 1, \quad \forall j \in J, \quad \forall t \in T_j. \tag{9}$$

$$e_{j,\,t} = s_{j,\,t} + p_{j,\,t}, \quad \forall j \in J. \tag{10}$$

Each task in a job must start only after the completion of its predecessor:

$$s_{j,\,t+1} \geq e_{j,\,t}, \quad \forall j \in J, \quad \forall t \in T_j \setminus \left\{t_{\max j}\right\}. \tag{11}$$

where $t_{\max j}$ is the last task of job j. If two distinct tasks are assigned to the same machine, they must be sequenced with the appropriate setup time:

$$s_k \geq e_i + s_{i,k} - M\left(3 - x_{i,m} - x_{k,m} - y_{i,k,m}\right), \tag{12}$$

$$s_i \geq e_k + s_{k,i} - M\left(3 - x_{i,m} - x_{k,m} - \left(1 - y_{i,k,m}\right)\right). \tag{13}$$

These inequalities enforce that if both tasks are assigned to m, then either i precedes k (first inequality active with setup $s_{i,k}$) or k precedes i (second inequality with $s_{k,i}$). The $C_{\max}$ is the completion time of the last finishing task across all jobs; it is enforced by

$$C_{\max} \geq e_{j,t},\ j \in J, \quad \forall t \in T_j \tag{14}$$

and the objective is:

$$\min C_{\max}. \tag{15}$$

2.6 *Metaheuristics*

Metaheuristics are powerful methods to address complex optimization problems characterized by high dimensionality, non-convex search spaces, and multiple interdependent constraints. They are particularly suitable when exact methods become computationally infeasible, as they balance global exploration with local exploitation to find near-optimal solutions within reasonable time frames [39–41].

2.6.1 *Memetic algorithm*

Schedules are represented as ordered lists of job-task-machine-duration and are decoded with sequence dependent setups, enforcing job precedences and machine availability to evaluate the makespan. The MA uses mixed initialization: half random and half based on the Shortest Processing Time (SPT) rule to balance diversity and quality; reproduction relies on tournament selection, a precedence preserving crossover, and mutations that either swap tasks across different jobs or reassign a task to an eligible alternative machine. A light local search refines offspring, while stagnation and a time limit act as stopping criteria. Schedules are exported and assessed with the F_{eq} metric in a window W.

2.6.1.1 *MA pseudocode*

```
1. Initialization
1.1 Read data and setup matrix; sort by (Job, Task).
1.2 Build initial population of size $P$:
- $P/2$ random chromosomes
- $P/2$ SPT-based chromosomes.
2. Generational loop ($g = 1 \dots G$ or until time limit)
2.1 For each chromosome:
- Decode with setups
- Compute makespan (fitness).
2.2 Update best solution and stagnation counter
- Stop if stagnation $\geq$ threshold.
2.3 Reproduction (until population size $P$):
```

```
a) Tournament selection → pick parents $p1$, $p2$
b) Precedence-preserving crossover → child
c) Mutation:
(i) Safe swap between jobs
(ii) Machine change to another eligible machine
d) Light local search on child.
2.4 Replace old population with offspring.
3. Output & metrics
3.1 Select best feasible individual
- Decode to full schedule (start/finish per operation with setups).
3.2 Build Comma-Separated Values (CSV): (job, task, machine, start, end)
- Compute equivalent fronts within window $W$.
3.3 Return:
- Makespan (hours)
- Feq
- Full schedule (visualize with Gantt chart).
```

2.6.2 *TS*

A schedule is likewise encoded as an ordered list and decoded with sequence dependent setups. Searches start from SPT respecting randomized seeds and improve the incumbent via two neighborhoods: (i) reassignment to an alternative eligible machine and (ii) a safe inter job swap that preserves intra job order. A bounded tabu list prevents cycling; an aspiration rule admits tabu moves if they improve the best so far. Periodic intensification (biased neighbor choice) and diversification (restart after stagnation) are applied. Each run exports the detailed schedule, a simplified timeline, and the equivalent-fronts score for window W.

2.6.2.1 *TS pseudocode*

```
1. Initialization
1.1 Read jobs and the sequence-dependency (setup) matrix.
1.2 Sort operations by (job, task).
1.3 Build data maps:
- Eligible machines per task
- Minimum task index per job.
1.4 Set parameters:
- Number of starts $S$
- Time limit per run
- Tabu size bounds
- Initial neighborhood size
- Stagnation threshold
- Aspiration probability
- Intensification interval.
1.5 Create $S$ SPT-respecting randomized seeds:
- Preserve task order within each job
- Assign each task to a random eligible machine.
2. Single-trajectory Tabu Search (per start)
Repeat until time limit or max iterations:
2.1 Decode current solution with setups → compute makespan.
2.2 Update best solution if improved.
```

```
2.3 Generate neighbors (size $k$):
- Machine change: reassign an operation to another eligible machine
- Safe swap: exchange two operations from different jobs, preserving per-job task order.
2.4 Admissibility check:
- Discard infeasible solutions
- Reject tabu moves unless:
(i) Strictly better than best-so-far (aspiration), or
(ii) Accepted with small aspiration probability.
2.5 Select best admissible neighbor (lowest makespan).
2.6 Move & update:
- Apply move
- Push move signature to bounded tabu list
- Update best solution if improved.
2.7 Adaptive neighborhood:
- Shrink $k$ on non-improvement
- Expand $k$ on improvement (within bounds).
2.8 Intensification:
- Every I iterations, bias machine-change toward promising candidates.
2.9 Diversification:
- If stagnation $\geq$ threshold, restart from fresh SPT-respecting seed and reset $k$.
3. Multi-start control
3.1 Run Tabu Search trajectory for each seed within per-run time budget.
3.2 Keep global best solution across all starts.
4. Output & metrics
4.1 Export best schedule with full timing (setup start/end; operation start/end).
4.2 Build timeline CSV: (job, task, machine, start, end).
4.3 Compute Equivalent Fronts on window $W$:
- Add official_timeline ("Time Oficial")
- Assign front_id and time_share
- Compute score = $\Sigma$ (front_id $\times$ time_share).
4.4 Return:
- Makespan (hours)
- Best schedule
- Feq value
- Per-run summary (over 30 runs).
```

2.6.3 *CMTS-PR*

The cooperative variant launches multiple tabu trajectories from greedy, setup-aware seeds (plus a minority of SPT respecting randomized seeds) and coordinates them through an elite set. Four neighborhoods are combined (same-machine insertion; reassign and insert; safe inter job swap; machine change) with reactive tabu tenure, adaptive neighborhood size, and a decaying frequency memory to discourage overused arcs and machine changes. Cooperation is further reinforced by path relinking, advancing from the current solution toward the most distant elite (machine alignment followed by safe swaps), accepting improving intermediates. Anytime curves are logged; the best schedule per run is exported along with the simplified timeline and F_{eq} metric.

2.6.3.1 *CMTS-PR pseudocode*

```
1. Parallel starts
  1.1 Build $N$ seeds:
    - Majority greedy setup-aware
    - Minority SPT-random.
  1.2 Initialize:
    - Best = $\infty$
    - Elite set E = $\varnothing$
    - Frequency memory $\mathscr{F}$.
2. Trajectories (per seed, until time or iteration limit)
  2.1 Decode current solution with setups $\longrightarrow$ compute makespan.
  2.2 Generate neighbors (adaptive size):
    - Same-machine insertion
    - Reassign-and-insert
    - Safe swap (preserve job order)
    - Machine change.
  2.3 Admissible pick:
    - Exclude tabu unless aspired
    - Compute score = makespan + freq-penalty ($\mathscr{F}$)
    - Select best neighbor.
  2.4 Update:
    - Apply move
    - Push move to bounded tabu list
    - Update $\mathscr{F}$ (move + schedule arcs) and apply decay.
  2.5 Elite cooperation:
    - If improved:
      * Update best
      * Insert into $E$ with diversity check
      * Reset stagnation
    - Else:
      * Increase tenure
      * Adapt neighborhood size (shrink/enlarge).
  2.6 Intensification/Path-Relinking (PR):
    - Every $I$ iterations, bias neighbor generation
    - If $E \neq \varnothing$ at PR interval:
      * Select guide $\in E$ (max machine-Hamming distance)
      * Path-relink current $\longrightarrow$ guide
      * Accept improving PR states.
  2.7 Diversification:
    - If stagnation $\geq$ threshold:
      * Restart from new greedy or SPT-random seed.
3. Finalization
  3.1 Return global best across all trajectories.
  3.2 Export schedule CSV + simplified timeline.
  3.3 Compute equivalent fronts over window $W$.
  3.4 Save anytime logs.
```

2.6.4 *NSGA-II with SDS*

We represent a schedule as an ordered list of triplets (job, task, machine). A decoder enforces job precedences and machine availability and inserts sequence dependent setups based on the previous operation run on each machine. Objective f_1 minimizes the makespan; objective f_2 maximizes early completion by counting tasks that finish before a fixed period. The algorithm follows the standard NSGA-II flow, with a precedence preserving crossover, safe mutations (inter-job swap and machine change), and a decode cache to speed up evaluations. Output include best schedules, Pareto front, and a time weighted F_{eq} metric computed from a simplified timeline.

2.6.4.1 *NSGA-II + SDS pseudocode*

```
1. Initialization
1.1 Set random seed.
1.2 Read jobs and sequence-dependency matrix.
1.3 Sort operations by (job, task).
1.4 Build initial population of size $P$:
- Fraction SPT-respecting (optionally with light machine noise)
- Remainder random (eligible machines assigned uniformly).
2. Generational loop (for $g = 1 \dots G$ or until time limit)
2.1 Decode each chromosome with setups → compute objectives:
- $f_1$: makespan of decoded schedule
- $f_2$: negative count of operations that finish within window $W$.
2.2 Apply fast non-dominated sorting → fronts $F0$, $F1$, $\dots$.
2.3 Compute crowding distance within each front.
2.4 Selection & variation (until new population size $P$):
- Tournament selection (rank first, then crowding)
- Precedence-preserving crossover
* Preserves intra-job order
* Inherits machines when valid
- Mutation:
* Safe inter-job swap
* Machine change to another eligible machine
- (Optional) light local improvement
* e.g., same-machine insertion if it reduces $f_1$.
2.5 Elitist replacement $(\mu + \lambda)$:
- Merge parents and offspring
- Sort by fronts
- Truncate by crowding to maintain diversity.
2.6 Track best Pareto front
- Stop early on prolonged stagnation or time limit.
3. Output & metrics
3.1 Export Pareto set $F0$.
3.2 Decode representative solutions:
- Best $f_1$
- Best $f_2$.
3.3 Build simplified timeline CSV:
- (job, task, machine, start, end).
3.4 Compute equivalent fronts in window $W$.
3.5 Return makespan, tasks-within-$W$, schedules, and Feq score.
```

3. Case study

3.1 *Experimental setup*

All methods (MILP, CP, and metaheuristics) were executed on a Personal Computer (PC) with 16 GB RAM, Intel(R) Core(TM) i5-8400T CPU @ 1.70 GHz, running Windows 10. For the metaheuristics, each run was limited to 60 seconds, with early termination if convergence occurred. Performance was evaluated in terms of makespan and the F_{eq}, using a time window of W = 780 minutes. The hyperparameter settings adopted for the metaheuristics are summarized in Table 1. Each metaheuristic run outputs the best schedule (full and simplified timelines), its F_{eq} score, and makespan. In addition, NSGA-II and CMTS-PR provide further outputs, such as Pareto fronts, anytime curves, and aggregate statistics over 30 runs.

Table 1. Hyperparameters of both case studies

Metaheuristics	Hyperparameter settings
TS	Multi-start with 700 SPT-respecting seeds; max 5,000 iterations or 60 s. Neighborhoods: (i) machine change, (ii) safe inter-job swap. Adaptive neighborhood size: 100 → [10, 50]. Tabu list: 10-30 moves, aspiration probability = 0.2. Intensification every 150 iterations; diversification after 100 stagnant iterations.
MA	Population = 500; max 1,000 generations or 60 s. Initialization: 50% random, 50% SPT-based. Tournament selection ($k = 3$). Precedence-preserving crossover. Mutation rate = 0.20 (safe swap or machine change). Local search ≤ 5 improving steps. Fitness = makespan via setup-aware decoder.
NSGA-II	Objectives: f_1 = makespan (min), $f_2 = -$(#operations ≤ T). Population = 150; max 200 generations or 60 s. Initialization: 40% SPT-respecting (half greedy, half noisy), 60% random. Selection: rank-based, crowding distance tie-break. Crossover: P3X precedence-preserving. Mutation rate = 0.25 (safe swap or machine change). Replacement: elitist ($\mu + \lambda$). Decode cache to speed evaluations.
CMTS-PR	60 parallel starts (mostly greedy setup-aware, some SPT-random). Max 10,000 iterations or 60 s. Neighborhoods: same-machine insertion, reassign-and-insert, safe inter-job swap, machine change. Reactive tabu tenure = 10-40; adaptive neighborhood size (start = 120). Aspiration probability = 0.20. Frequency memory: decay = 0.997, penalty $\lambda = 0.0015$. Intensification every 60 iterations; path-relinking every 80 iterations.

3.2 *First case study*

Table 2. Mining cycle activities in first case study

No.	Activity	Duration (min)	Fraction	Equipment
1	Mucking	156	0.15	3
2	Bolt drilling	148	0.14	3
3	Grouting	151	0.14	2
4	Mesh installation	135	0.13	2
5	Bolting	148	0.14	2
6	Shotcreting	125	0.12	2
7	Drilling	202	0.19	3
	Total	1,065	1.00	

The mine has a total of 21 UHW located at the production level of a mine exploited using the Panel Caving method. The operation is organized into two 12-hour shifts, with an effective time of 6.5 hours per shift. Blasting is carried out exclusively during the night shift. The activities corresponding to the mining cycle, along with their respective durations, are detailed in Table 2. The initial status of the UHW at the start of the shift is presented in Table 3. In the operation cycle, mucking involves the removal of extracted material and its transportation out of the work area. Bolt drilling refers to drilling holes in the walls to install support bolts. Grouting involves injecting cementing material into the drilled holes to secure the bolts and improve stability. Mesh installation is the placement of metal mesh to reinforce the walls and prevent rock falls. Bolting refers to the installation of anchor bolts for additional structural support. Shotcreting involves spraying concrete onto surfaces to stabilize and reinforce the terrain. Finally, drilling consists of drilling the advance face to place explosives and perform rock blasting.

Table 3. UHW activities at the start of the shift in first case study

UHW	Activity	UHW	Activity
1	Bolt drilling	12	Bolt drilling
2	Bolt drilling	13	Mucking
3	Shotcreting	14	Mucking
4	Shotcreting	15	Shotcreting
5	Drilling	16	Mucking
6	Bolting	17	Bolting
7	Bolt drilling	18	Bolting
8	Mucking	19	Bolt drilling
9	Bolting	20	Mucking
10	Grouting	21	Mucking
11	Bolting		

3.2.1 *Scenario 0*

In the first case study, we benchmark an exact MILP against a CP model while varying the number of UHW fronts in Table 4. To isolate modeling effects, sequence-dependent setups are fixed at 0, and the instance includes 99 operations. We report (Makespan, F_{eq}) for each UHW level; where both methods return solutions, CP matches MILP's objective values, and we then push UHW further to probe scalability.

Table 4. CP vs. MILP across UHW fronts (first case study; setups = 0 min; 99 tasks)

UHW	MILP (Makespan/F_{eq})	CP (Makespan/F_{eq})	UHW	MILP (Makespan/F_{eq})	CP (Makespan/F_{eq})
1	15.15 h/0.7324	15.15 h/0.7324	12	20.18 h/6.0920	20.18 h/6.0920
2	15.15 h/1.4648	15.15 h/1.4648	13		21.03 h/6.6845
3	15.15 h/1.7718	15.15 h/1.7718	14		22.70 h/7.1380
4	15.15 h/2.0789	15.15 h/2.0789	15		22.70 h/7.4451
5	15.15 h/2.2685	15.15 h/2.2685	16		23.55 h/7.8870
6	15.15 h/2.7146	15.15 h/2.7146	17		23.62 h/8.3086
7	17.67 h/3.3662	17.67 h/3.3662	18		25.22 h/8.5959
8	18.52 h/4.0357	18.52 h/4.0357	19		26.08 h/8.7612
9	18.52 h/4.4141	18.52 h/4.4141	20		26.08 h/9.0701
10	18.52 h/5.1953	18.52 h/5.1953	21		26.08 h/9.355
11	18.52 h/5.6413	18.52 h/5.6413			

On our hardware, MILP solved UHW ≤ 10 in < 2 s and UHW = 12 in ~1 h, but for UHW ≥ 13 it did not return a solution within 3 hours. In contrast, CP attained the same optimal (Makespan, F_{eq}) values as MILP for UHW ≤ 12 and continued to deliver solutions beyond that point: ≤ 3 s up to UHW = 13, ≈ 200 s for UHW = 14-19, then increasing to ~ 3,600 s at UHW = 20 and ~ 3 h at UHW = 21. This highlights CP's stronger scalability on this instance family while preserving optimality where MILP is tractable.

3.2.2 *Scenario I*

In this first scenario, the equipment travel time between UHW is 0 minutes, which is not realistic, although this assumption is not realistic, it allows for isolating and assessing the specific impact of including travel time in short-term planning. The mathematical model provides an optimal solution of $F_{eq} = 9.35$, calculate. Figure 3 shows the results obtained through TS, MA, NSGA-II, and CMTS-PR. When comparing these results with CP, it is observed that CMTS-PR and NSGA-II has the no difference compared to the optimal result generated by the mentioned models.

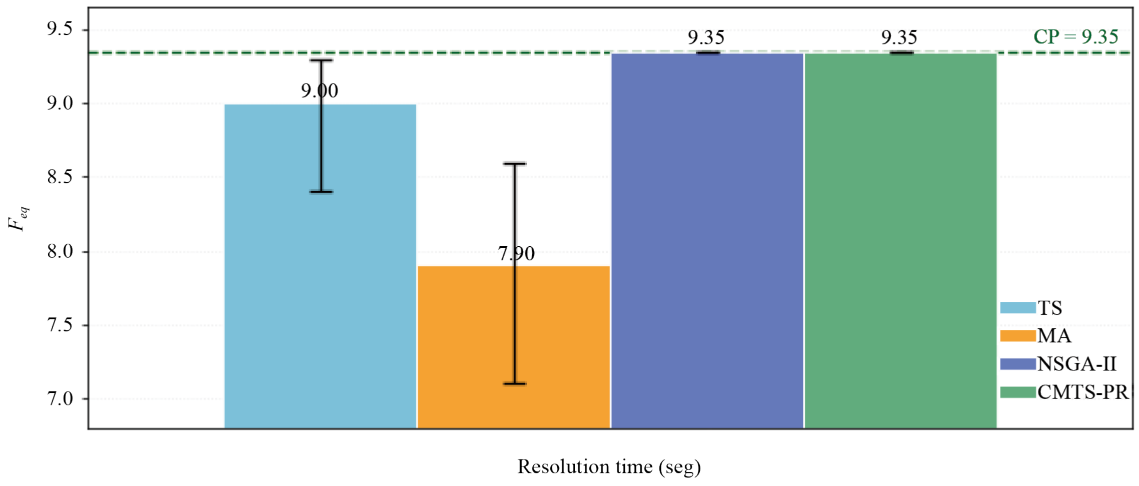

Figure 3. Equivalent underground headings index by scenario I (0 min) in first case study

3.2.3 *Scenario II*

In this second scenario, the transfer time is set at a constant 15 minutes for all equipment movements between UHW, regardless of the actual distances. Although the CP model provides the best outcome, it also generates an operationally feasible solution with an optimal value of $F_{eq} = 8.56$, calculated in under a minute. Figure 4 displays the resulting schedule.

Figure 5 presents the results obtained using TS, MA, NSGA-II, and CMTS-PR. When compared to the operational results of the CP optimization model, it is observed that CMTS-PR and NSGA-II achieves the same result, demonstrating its strong capability to solve the planning problem more efficiently in terms of time. Figure 6 complements these findings by showing the detailed schedule generated by the TS method, confirming that the solution is operationally feasible and respects equipment sequencing and setup times.

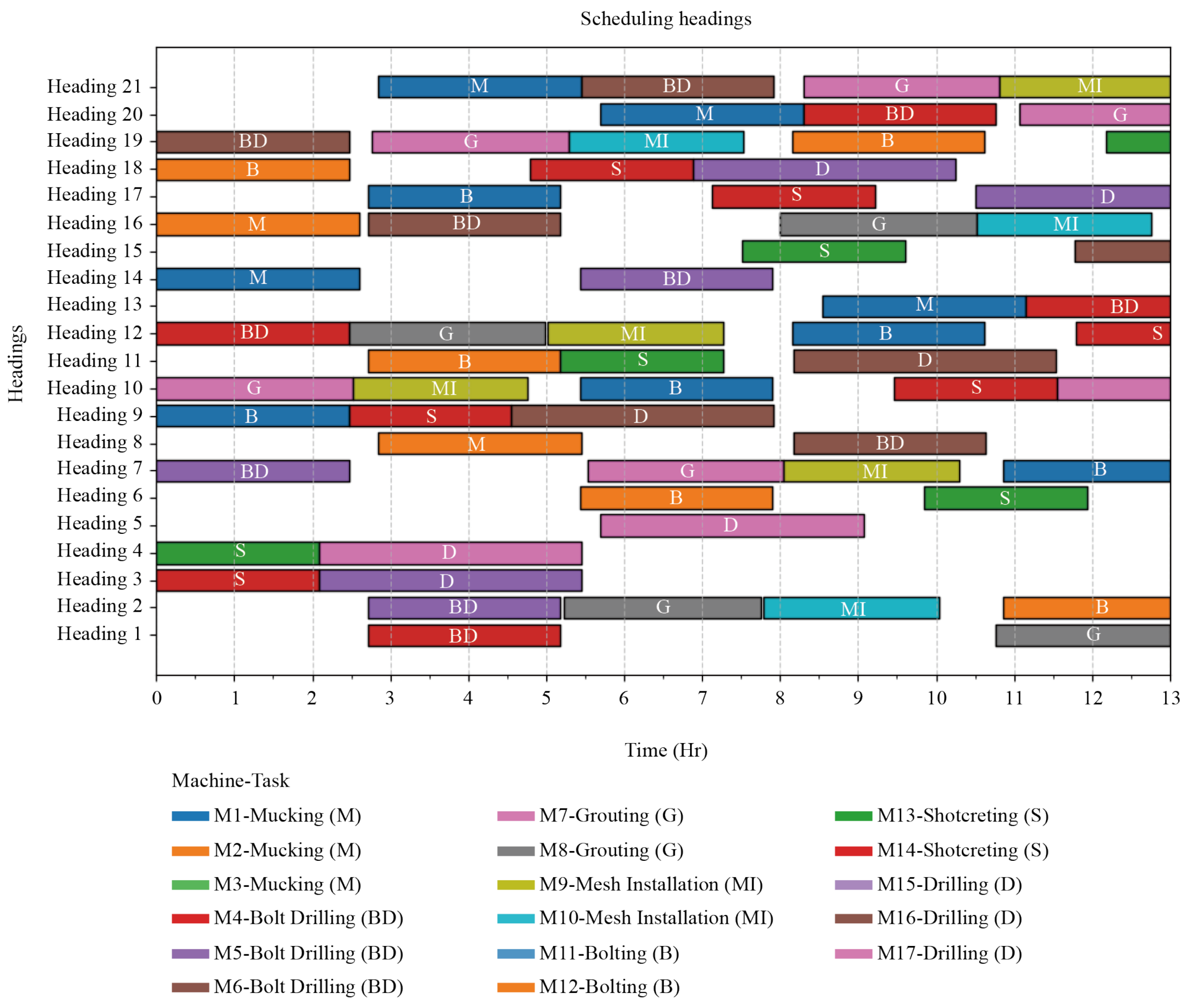

Figure 4. Scheduling by CP model for scenario I in first case study

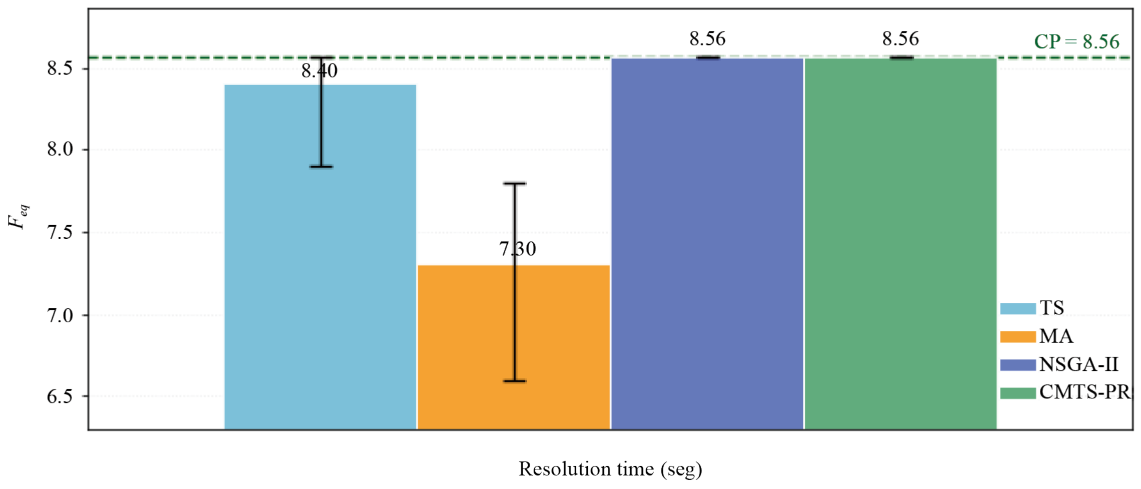

Figure 5. Equivalent underground headings index by scenario II (15 min) in first case study

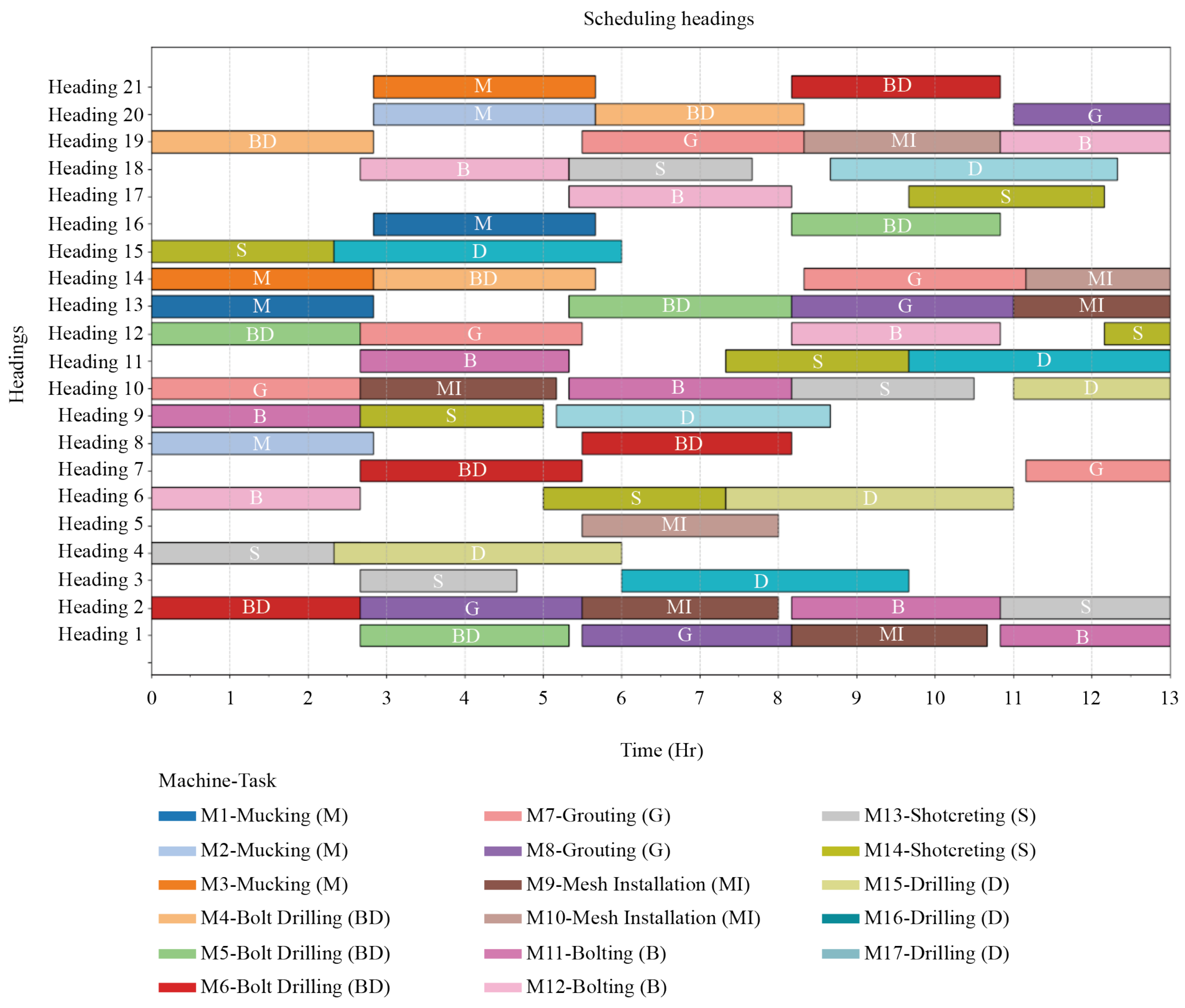

Figure 6. Scheduling by TS method for scenario II in first case study

3.2.4 *Scenario III*

In Scenario 3, variable travel times between UHW are introduced, ranging from 10 to 60 minutes. Due to this variability, it is no longer feasible to include travel time as an additional value to the operating time of mining cycle activities. Figure 7 shows the results obtained with TS, MA, NSGA-II, and CMTS-PR. Figure 8 presents the schedule corresponding to the best solution generated by TS. Furthermore, this schedule is operational, as it ensures that each activity starts and ends with the same assigned equipment.

In summary, in this first application case, CMTS-PR consistently outperforms MA across all scenarios, achieving solutions equivalent to those obtained by the CP model but with considerably lower runtimes. Similarly, NSGA-II delivers results of the same excellence as CMTS-PR, while TS also shows better performance than MA, consolidating itself as a more robust alternative among the metaheuristics evaluated.

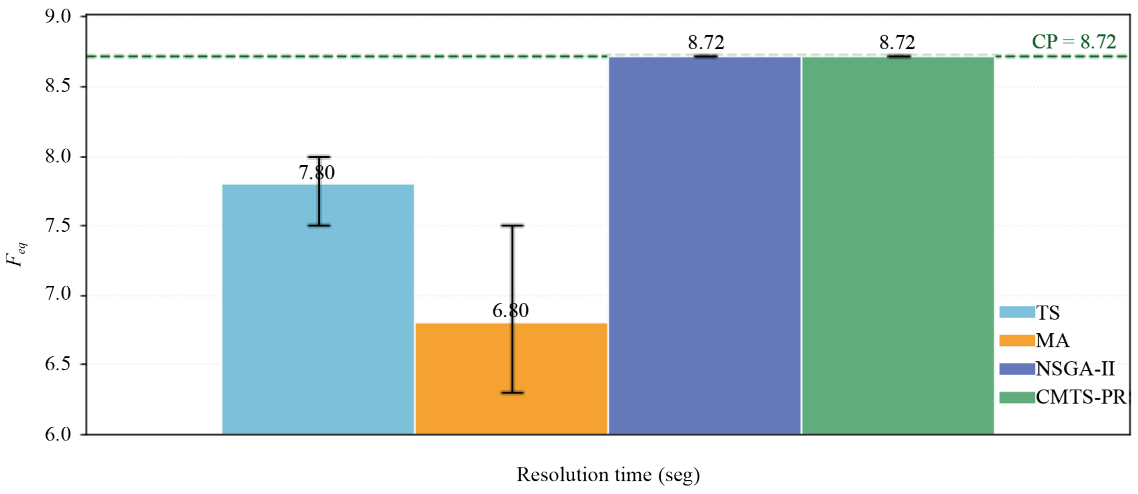

Figure 7. Equivalent underground headings index by scenario III (10-60 min) in first case study

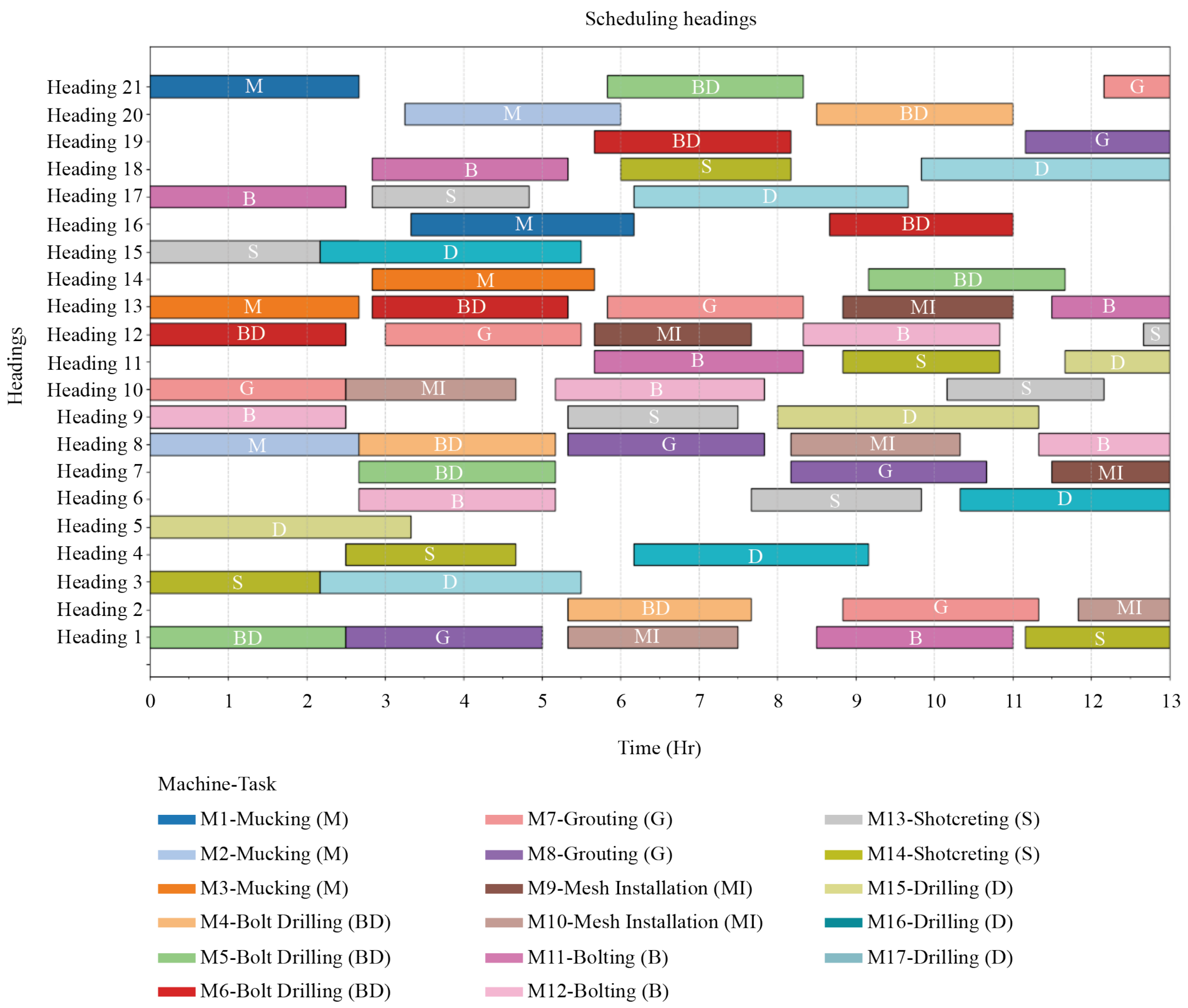

Figure 8. Scheduling by TS method of scenario III in first case study

3.3 *Second case study*

In this second case study, data is available for 3 work shifts and 16 UHW, all planned by an expert short-term planning engineer. This process is classified as manual planning. The initial state of each shift is detailed in Table 5. The mine considers two shifts of 12 hours, with a total of 10 effective hours per shifts. The activities for this case and their lengths and available equipment are presented in Table 6. The planning team's primary objective is to ensure that at least four work fronts are ready for blasting by the end of each shift. Travel time between work fronts varies depending on the distance, ranging from a minimum of 10 minutes to a maximum of 210 minutes.

Table 5. Initial state and operation's requirement per from second case study. Mucking (Mu), Cleaning (Cl), Scaling (Sc), Pre-Shotcreting (PSH), Bolt Drilling or Drilling Support (DS), Grouting (Gr), Meshing (ME), Bolting or Hilting (Hi), Drilling development (DD), Shotcreting (SH), Charging (CH)

Shift	Work heading															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Shift 1	CH	SH	SH	Hi	ME	ME	L	C	Gr	DS						
Shift 2	DD	DD	CH	DD	Gr	Cl	ME	Gr	C	DS	DS					
Shift 3	Gr	CH	H	DD	DD	ME	Gr	ME	ME	DS	PSH	Mu	Mu	DS		

Table 6. HW activities at the start of the shift in second case study

No.	Activity	Duration (min)	Fraction	Equipment
1	Mucking	120	0.12	4
2	Cleaning	60	0.06	1
3	Scaling	30	0.03	1
4	Pre-shotcreting	30	0.03	1
5	Drilling support	180	0.18	2
6	Grounting	90	0.09	1
7	Meshing	90	0.09	1
8	Hilting	90	0.09	1
9	Drilling development	180	0.18	3
10	Shotcreting	30	0.03	1
11	Charging	90	0.09	1
	Total	990	1	

The results of the first shift evaluated have an average of $F_{eq} = 4.06$ using CMTS-PR, which means an increase of 103% compared to manual programming, and 4.67% lower than the optimum found by CP model like an optimal solution (Figure 9). It is essential to highlight that although CMTS-PR achieves the best results, this method becomes exponentially more computationally demanding and slower in execution time as the number of fronts and equipment increases. Therefore, CMTS-PR is largely viable in practical terms as a supportive tool for the daily tasks of a short-term planning engineer.

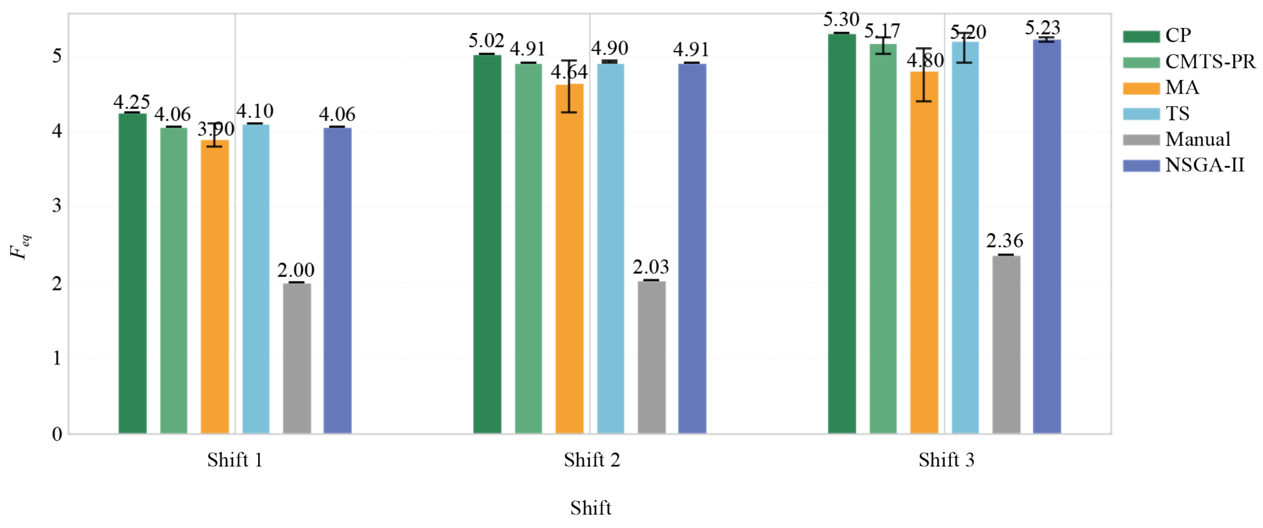

Figure 9. Equivalent underground headings index per shifts in second study case

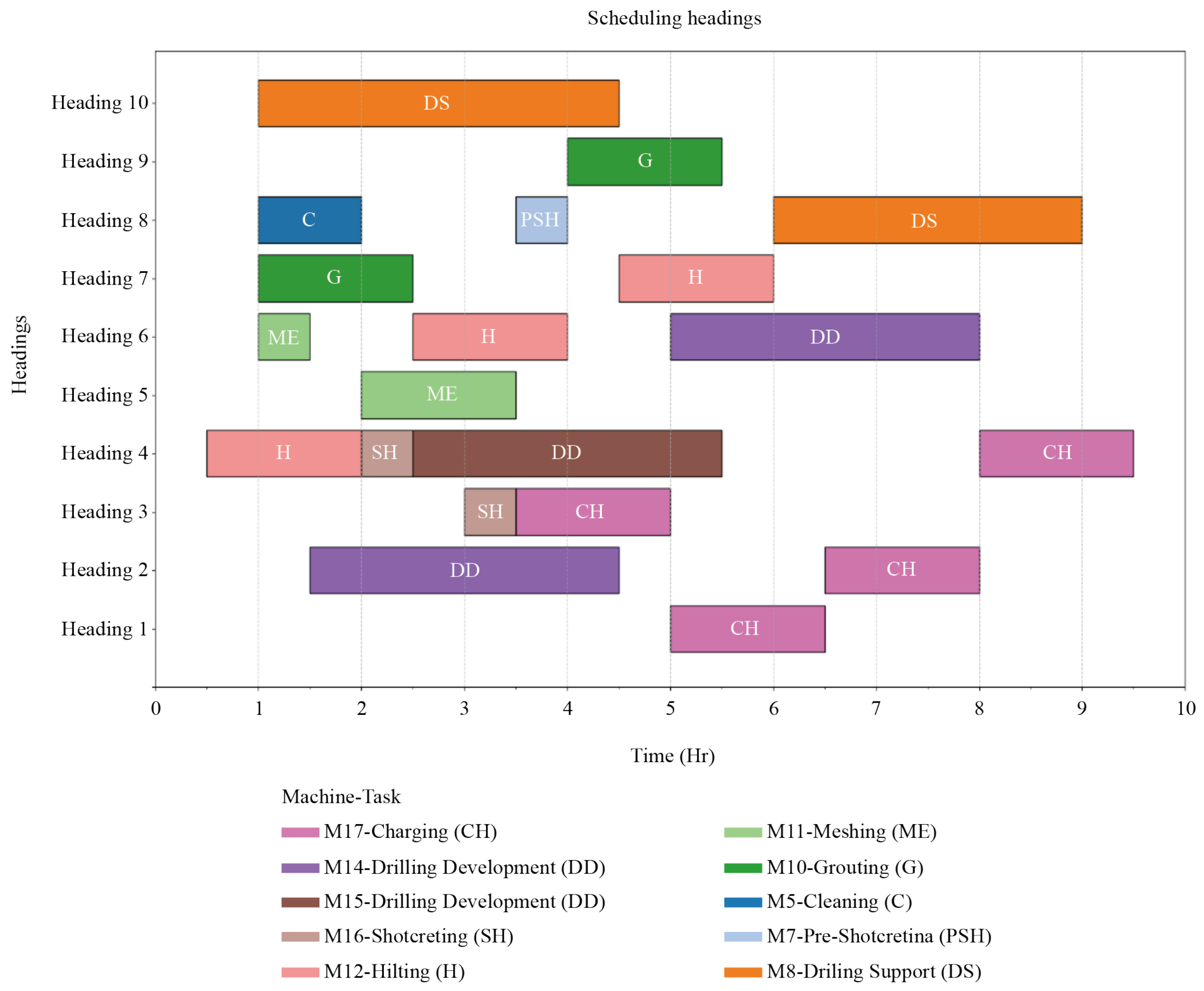

Figure 10. Scheduling shift one by engineer expert in second case study

The scheduling using CMTS-PR results are operationally more efficient, as they optimize the coordination of equipment for various unit operations, surpassing the manual scheduling performed by an experienced engineer (see Figure 10). The activity durations, the sequence of operations within a single work heading, and the equipment travel times between different UHW are all respected, achieving significantly shorter resolution times compared to manual planning.

4. Discussion

Short-term underground scheduling is still commonly executed with manual or spreadsheet based practices that struggle to guarantee optimality or consistency [21–25]. In parallel, recent work shows that exact and hybrid optimization can capture the true operational drivers in underground mines [2, 17, 18, 22, 24]. Our study advances this line by formulating a flexible job-shop with unrelated parallel equipment and sequence-dependent setups (to encode travel and transfer times), a combination still uncommon in underground contexts.

4.1 *Prioritization of times between headings and sequence-dependent transfer times*

Transfer times between headings are often simplified as constants or omitted in short-term models [17, 18, 21], despite their strong impact on feasibility and shift productivity. By treating transfers as sequence-dependent setup times, our formulation directly captures spatial and logistical effects. Under this setting, our TS family, particularly CMTS-PR, delivers optimal or near optimal solutions in the case studies, including the realistic scenario with variable travel times (10-60 min) where CP becomes time consuming.

4.2 *Heterogeneous equipment and computational complexity*

Modeling unrelated equipment reflects heterogeneous capabilities and availabilities seen in practice [24–29], but it raises complexity, FJSSP is NP-hard. In such regimes, TS is well-suited: decades of evidence show it scales effectively on hard shop-scheduling variants [36, 37, 41]. Consistent with that literature, our CMTS-PR attains high quality schedules within strict time budgets, while CP and MILP become challenging as problem size and routing interplay grow [2, 8].

4.3 *Exact baselines and scalability*

On instances with zero/constant setups, CP and MILP return identical optima for moderate UHW levels (see Table 3), but as UHW increases MILP fails to produce solutions within a practical 3-hour window, whereas CP still reaches the same objective values and extends further before hitting time limits. These observations align with prior reports on the scalability trade-offs of exact methods in mine scheduling [2, 12, 26, 30] and motivate time capped metaheuristics for day-to-day planning.

4.4 *Metaheuristic baselines*

MA and NSGA-II are established choices for flexible shops [39, 42, 43] and were competitive under the same time caps in our tests. However, the CMTS-PR design, multi-start intensification, adaptive neighborhoods, frequency memory, and path-relinking, consistently matches CP when setups are simple and outperforms it in the realistic, sequence-dependent regime, delivering operational schedules where CP struggles within viable wall-times [8, 36, 39, 41].

4.5 *Practical implications and limits*

For daily underground planning, a method must deliver feasible, high quality schedules under tight time budgets [17]. By integrating sequence-dependent travel and heterogeneous fleets, our approach narrows the gap between optimization and operations [7, 10, 18]. While we do not explicitly model stochastic failures, the formulation and algorithm are compatible with uncertainty extensions (e.g., scenario sampling or robust penalization) proposed in recent mining optimization studies [4, 9, 15].

5. Conclusions

This study confirms the viability and effectiveness of metaheuristic approaches, particularly CMTS-PR, for optimizing the scheduling of development and preparation activities in underground mining. Across two case studies, CMTS-PR consistently produced operationally feasible schedules with performance comparable to CP while requiring significantly less computational time, demonstrating its practical applicability for real-world scenarios. In the first case study, involving 21 work fronts and three distinct scenarios, TS matched the results of CP in scenarios with no or constant transfer times, and crucially, outperformed it in the most realistic setting with variable travel times (10-60 minutes). In that scenario, CP did not return optimal solutions within a viable time window for a mining project, with runs reaching the 3-hour mark without delivering an operationally usable optimum. This highlights CMTS-PR ability to not only approximate optimal results but also adapt better to the variability inherent in underground operations, achieving solutions within 2.94% of CP in the most complex scenario. Furthermore, CMTS-PR accomplished this in less than 60 seconds, underscoring its computational efficiency and suitability for real-time decision-making. In the second case study, involving 3 shifts and 16 work fronts planned by an expert engineer, manual planning showed the weakest performance. TS outperformed GA in all instances and closely matched CP, achieving improvements of up to 120% in the equivalent underground headings index relative to manual planning. Moreover, CMTS-PR delivered schedules with minimal deviation from CP but within a fraction of the runtime, confirming its practical value for daily decision-making Overall, the findings highlight CMTS-PR as a robust and scalable tool that bridges the gap between exact optimization methods, which often lack scalability, and manual scheduling, which is limited by subjectivity and inefficiency. By providing high quality schedules within operational timeframes, the proposed approach advances the semi-automation of short-term mine planning and offers a reliable foundation for extending optimization to longer horizons, such as weekly or monthly scheduling. The ability of CMTS-PR to handle variable transfer times, heterogeneous equipment, and operational uncertainty positions it as a versatile and practical solution for improving resource utilization and operational continuity in underground mining. This study lays the groundwork for more advanced methodologies that can be applied across different mining methods and planning horizons, ultimately contributing to more efficient and sustainable mining operations. Finally, as it can be deducted from the Funding section of this manuscript, this research has raised the interest of other sectors, such as the steel industry, who wishes to maximize the efficiency of all its processes in terms of both industrial and social impact.

Funding

Work funded by project SILENCE—European Commission—Research Program of the Research Funds for Coal and Steel—Prj. No.: 101112516. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

Conflict of interest

The authors declare no competing financial interest.

References

[1] Brazil M, Grossman PA, Lee DH, Rubinstein JH, Thomas DA, Wormald NC. Decline design in underground mines using constrained path optimisation. *Transactions of the Institutions of Mining and Metallurgy, Section A: Mining Technology*. 2008; 117(2): 93-99. Available from: https://doi.org/10.1179/174328608X362668.

[2] Newman AM, Rubio E, Caro R, Weintraub A, Eurek K. A review of operations research in mine planning. *Interfaces*. 2010; 40(3): 175-252. Available from: https://doi.org/10.1287/inte.1090.0492.

[3] Mishchenko K, Åstrand M, Molander M, Lindkvist R, Viklund T. Developing a tool for automatic mine scheduling. In: *Proceedings of the 28th International Symposium on Mine Planning and Equipment Selection*. Cham: Springer; 2020. p.146-153. Available from: https://doi.org/10.1007/978-3-030-33954-8_18.

[4] Morales N, Mancilla D, Miranda R, Vallejos J. Optimal drift and level design in underground mining extracted by sublevel open stoping method. *SSRN Electronic Journal*. 2021. Available from: https://doi.org/10.2139/ssrn.3928373.

[5] Rahimi B, Sharifzadeh M, Feng XT. A comprehensive underground excavation design (CUED) methodology for geotechnical engineering design of deep underground mining and tunneling. *International Journal of Rock Mechanics and Mining Sciences*. 2021; 143: 104684. Available from: https://doi.org/10.1016/j.ijrmms.2021.104684.

[6] Diering T. Quadratic programming applications to block cave scheduling and cave management. In: *6th International Conference & Exhibition on Mass Mining*. Santiago, Chile; 2012.

[7] Nehring M, Topal E, Kizil M, Knights P. Integrated short- and medium-term underground mine production scheduling. *Journal of the Southern African Institute of Mining and Metallurgy*. 2012; 112(5): 365-378.

[8] O'Sullivan D, Newman A. Optimization-based heuristics for underground mine scheduling. *European Journal of Operational Research*. 2015; 241(1): 248-259. Available from: https://doi.org/10.1016/j.ejor.2014.08.020.

[9] Furtado e Faria M, Dimitrakopoulos R, Lopes Pinto CL. Integrated stochastic optimization of stope design and long-term underground mine production scheduling. *Resources Policy*. 2022; 78: 102918. Available from: https://doi.org/10.1016/j.resourpol.2022.102918.

[10] Rocher W, Rubio E, Morales N. Eight-dimensional planning-construction of an integrated model for mine planning involving constructability. In: *35th APCOM Symposium-Application of Computers and Operations Research in the Minerals Industry, Proceedings*. NSW; 2011.

[11] Terblanche S, Bley A. An improved formulation of the underground mine scheduling optimisation problem when considering selective mining. *ORiON*. 2015; 31(1). Available from: https://doi.org/10.5784/31-1-422.

[12] Muñoz G, Espinoza D, Goycoolea M, Moreno E, Queyranne M, Letelier OR. A study of the Bienstock-Zuckerberg algorithm: applications in mining and resource constrained project scheduling. *Computational Optimization and Applications*. 2018; 69(2): 501-534. Available from: https://doi.org/10.1007/s10589-017-9946-1.

[13] Nancel-Penard P, Morales N, Rojas V, González T. A heuristic approach for scheduling activities with 'or'-precedence constraints at an underground mine. *International Journal of Mining, Reclamation and Environment*. 2020; 34(10): 748-762. Available from: https://doi.org/10.1080/17480930.2020.1734152.

[14] Blom M, Pearce AR, Stuckey PJ. Short-term planning for open pit mines: a review. *International Journal of Mining, Reclamation and Environment*. 2019; 33(5): 318-339. Available from: https://doi.org/10.1080/17480930.2018.1448248.

[15] Nelis G, Morales N, Jelvez E. Optimal mining cut definition and short-term open pit production scheduling under geological uncertainty. *Resources Policy*. 2023; 81: 103340. Available from: https://doi.org/10.1016/j.resourpol.2023.103340.

[16] Chowdu A, Goycoolea M, Brickey A. Mine schedule optimization and mine operational realities: bridging the gap. In: *Mining Goes Digital-Proceedings of the 39th International Symposium on Application of Computers and Operations Research in the Mineral Industry*. APCOM; 2019.

[17] Chowdu A, Nesbitt P, Brickey A, Newman AM. Operations research in underground mine planning: a review. *INFORMS Journal on Applied Analytics*. 2022; 52(2): 109-231. Available from: https://doi.org/10.1287/inte.2021.1087.

[18] Campeau LP, Gamache M. Short-term planning optimization model for underground mines. *Computers and Operations Research*. 2020; 115: 104642. Available from: https://doi.org/10.1016/j.cor.2019.02.005.

[19] Topal E. Early start and late start algorithms to improve the solution time for long-term underground mine production scheduling. *Journal of the Southern African Institute of Mining and Metallurgy*. 2008; 108(2): 99-107.

[20] Cortés D, Martinez Y, Silva Balocchi M. Maximising resource utilisation in scheduling of underground mining works with multiple faces. In: *Caving 2018: Proceedings of the Fourth International Symposium on Block and Sublevel Caving*. Australian Centre for Geomechanics; 2018.

[21] Andrade AB, Rampazzo PCB. Understanding plan's priorities: short term scheduling optimization. In: *Mining Goes Digital-Proceedings of the 39th International Symposium on Application of Computers and Operations Research in the Mineral Industry*. APCOM; 2019.

[22] Aalian Y, Pesant G, Gamache M. Optimization of short-term underground mine planning using constraint programming. *Leibniz International Proceedings in Informatics*. 2023; 280: 1-16. Available from: https://doi.org/10.4230/LIPIcs.CP.2023.6.

[23] Little J, Knights P, Topal E. Integrated optimization of underground mine design and scheduling. *Journal of the Southern African Institute of Mining and Metallurgy*. 2013; 113(10): 775-785.

[24] Åstrand M, Johansson M, Greberg J. Underground mine scheduling modelled as a flow shop: a review of relevant work and future challenges. *Journal of the Southern African Institute of Mining and Metallurgy*. 2018; 118(12). Available from: https://doi.org/10.17159/2411-9717/2018/v118n12a5.

[25] Toledo AAT, Marques DM, Costa JFCL, Capponi LN. Short-term mine scheduling targeting stationary grades. *REM-International Engineering Journal*. 2022; 75(1). Available from: https://doi.org/10.1590/0370-44672020750134.

[26] Newman AM, Kuchta M. Using aggregation to optimize long-term production planning at an underground mine. *European Journal of Operational Research*. 2007; 176(2): 1205-1218. Available from: https://doi.org/10.1016/j.ejor.2005.09.008.

[27] Song Z, Schunnesson H, Rinne M, Sturgul J. Intelligent scheduling for underground mobile mining equipment. *PLoS ONE*. 2015; 10(6). Available from: https://doi.org/10.1371/journal.pone.0131003.

[28] Åstrand M, Johansson M, Zanarini A. Fleet scheduling in underground mines using constraint programming. In: *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Cham: Springer; 2018. p.605-613. Available from: https://doi.org/10.1007/978-3-319-93031-2_44.

[29] Schulze M, Rieck J, Seifi C, Zimmermann J. Machine scheduling in underground mining: an application in the potash industry. *OR Spectrum*. 2016; 38(2): 365-403. Available from: https://doi.org/10.1007/s00291-015-0414-y.

[30] Martinez MA, Newman AM. A solution approach for optimizing long- and short-term production scheduling at LKAB's Kiruna mine. *European Journal of Operational Research*. 2011; 211(1): 184-197. Available from: https://doi.org/10.1016/j.ejor.2010.12.008.

[31] Wang H, Tenorio V, Li G, Hou J, Hu N. Optimization of trackless equipment scheduling in underground mines using genetic algorithms. *Mining, Metallurgy and Exploration*. 2020; 37(5): 1531-1544. Available from: https://doi.org/10.1007/s42461-020-00285-8.

[32] Miao W, Zhao X. Traffic congestion scheduling for underground mine ramps based on an improved genetic scheduling algorithm. *Applied Sciences*. 2024; 14(21): 9862. Available from: https://doi.org/10.3390/app14219862.

[33] Burmeister SC, Guericke D, Schryen G. Correction: a memetic NSGA II for the multi objective flexible job shop scheduling problem with real time energy tariffs. *Flexible Services and Manufacturing Journal*. 2024; 36(4): 1571-1573. Available from: https://doi.org/10.1007/s10696-023-09530-w.

[34] Lamghari A, Dimitrakopoulos R. A diversified tabu search approach for the open-pit mine production scheduling problem with metal uncertainty. *European Journal of Operational Research*. 2012; 222(3): 642-652. Available from: https://doi.org/10.1016/j.ejor.2012.05.029.

[35] Mousavi A, Kozan E, Liu SQ. Comparative analysis of three metaheuristics for short-term open pit block sequencing. *Journal of Heuristics*. 2016; 22(3): 301-329. Available from: https://doi.org/10.1007/s10732-016-9311-z.

[36] Brandimarte P. Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations Research*. 1993; 41(3): 157-183. Available from: https://doi.org/10.1007/BF02023073.

[37] Hurink J, Jurisch B, Thole M. Tabu search for the job-shop scheduling problem with multi-purpose machines. *Operations-Research-Spektrum*. 1994; 15(4): 205-215. Available from: https://doi.org/10.1007/BF01719451.

[38] Katoch S, Chauhan SS, Kumar V. A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications*. 2021; 80(5): 8091-8126. Available from: https://doi.org/10.1007/s11042-020-10139-6.

[39] Bianchi L, Dorigo M, Gambardella LM, Gutjahr WJ. A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing*. 2009; 8(2): 239-287. Available from: https://doi.org/10.1007/s11047-008-9098-4.

[40] Rajwar K, Deep K, Das S. An exhaustive review of the metaheuristic algorithms for search and optimization: taxonomy, applications, and open challenges. *Artificial Intelligence Review*. 2023; 56(11): 13187-13257. Available from: https://doi.org/10.1007/s10462-023-10470-y.

[41] Sörensen K, Sevaux M, Glover F. A history of metaheuristics. In: *Handbook of Heuristics*. Springer; 2018. Available from: https://doi.org/10.1007/978-3-319-07124-4_4.

[42] Pezzella F, Morganti G, Ciaschetti G. A genetic algorithm for the flexible job-shop scheduling problem. *Computers and Operations Research*. 2008; 35(10): 3202-3212. Available from: https://doi.org/10.1016/j.cor.2007.02.014.

[43] Talbi EG. *Metaheuristics: From Design to Implementation*. John Wiley & Sons, Inc.; 2009. Available from: https://doi.org/10.1002/9780470496916.